UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR
INGENIERO INDUSTRIAL

PROYECTO FIN DE CARRERA

RECONOCIMIENTO DE LOCALIZACIONES MEDIANTE MÁQUINAS DE SOPORTE VECTORIAL

AUTOR: VÍCTOR RIOBÓ OTERO

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En la actualidad la inteligencia artificial está presente en prácticamente todos los campos de la vida y aunque es u concepto relacionado con el software es relacionada por todo el mundo con la robótica. El desarrollo de los microprocesadores y de la informática ha posibilitado que las máquinas tengan cierta conciencia de sí mismas y puedan aprender de los actos que realizan para mejorar los procesos para los que fueron construidas.

Uno de las especialidades de la inteligencia artificial es el reconocimiento automático, herramientas que permiten gracias a sensores adquirir datos del entorno y reconocerlo. Las formas de abordar el reconocimiento automático son múltiples, una de ellas es mostrarle ejemplos a nuestro sistema (sea un robot, sistema informático…) indicando cuáles de ellos son los deseables y cuales los que no lo son. A base de entrenarlo con ejemplos el sistema se hará una idea o modelo y en el futuro podrá enfrentarse a una situación real y compararla con lo aprendido para saber cómo reaccionar.

Esta rutina se conoce como aprendizaje supervisado y es el objetivo de este proyecto. Realizar un sistema informático que simule a un robot que se pueda desplazar por un pasillo reconociendo su emplazamiento. Después de una etapa de aprendizaje se comprobará el grado de fiabilidad en el reconocimiento de las localizaciones para validar el algoritmo.

Dentro del concepto de aprendizaje supervisado existen diferentes algoritmos que parten de planteamientos lógicos y matemáticos distintos, siendo quizás el más empleado el de las redes neuronales que simulan la sinapsis del cerebro. En este caso se empleará el de las máquinas de soporte vectorial (SVM, del inglés support vector machine) tipo clasificación.

El desarrollo teórico de los distintos métodos de reconocimiento automático y aprendizaje supervisado se explica en el capítulo 2 donde se analizan las diferencias, similitudes y la formulación matemática, para entender los parámetros que afectan a cada algoritmo.

Una vez se ha optado por las máquinas de soporte vectorial como base del proyecto, se decide emplear Matlab como soporte informático y la librería Libsvm como algoritmo de aprendizaje y clasificación de SVM.

Paralelamente al desarrollo de una interfaz gráfica y al código informático se probó las diferentes opciones de parametrización que tienen las SVM, que van desde la elección de un kernel para la operación interna de datos (y obtención de los vectores soporte) a valores de márgenes y desviaciones.

Con el soporte informático y los algoritmos correctamente parametrizados se ensaya con y sin ruido y se valora las debilidades y las fortalezas de la formulación.

La parte más importante y objetivo final de un proyecto teórico son las conclusiones ya que aunque los resultados empíricos no sean los deseados, siempre se abren nuevas vías de investigación.

En este caso los resultados de las simulaciones fueron positivos y de ellos se extraen conclusiones en el capítulo 5, así como puntos a mejorar de la actual programación y nuevos enfoques que podrían mejorar la robustez y fiabilidad.

Finalizando el proyecto aparecen los anexos. Es el espacio reservado a mostrar tabulados los resultados numéricos y las guías de uso de las funciones empleadas en Matlab, tanto las librerías de algoritmos como el código desarrollado en exclusiva para el presente trabajo.

ABSTRACT

Nowadays, the artificial intelligence is present in virtually all fields of life and although it is a software related concept, it is associated worldwide with robotics. The development of microprocessors and computer technology has enabled that machines have some awareness of themselves and can learn from the acts done to improve the processes for which they were built.

One of the specialties of the artificial intelligence is the automatic recognition, tool that allows through sensors to acquire data in the environment and recognize it. The approaches to the automatic recognition are many, one of which is to show examples to our system (a robot, computer system ...) indicating which of them are desirable and those which are not. By training it with examples, the system will have an idea or model and in the future it will face a real situation and compare it with what it has learned to know how to react.

This routine is known as supervised learning and it is the target of this project. Perform a computer system that simulates a robot that can move down a hallway recognizing its location. After a learning period it shall be tested for reliability in the recognition of the locations to validate the algorithm.

Within the concept of supervised learning there are differents algorithms that start from different logical and mathematical approaches, perhaps the most used the neural networks that mimic the brain's synapses. In this case it shall be used support vector machines (SVM) classification type.

The theoretical development of the different methods of automatic recognition and supervised learning is explained in Chapter 2 where we analyze the differences, similarities and the mathematical formulation, to understand the parameters that affect each algorithm.

Once you have chosen support vector machines as base of the project, we decided to use Matlab as a carrier medium and the LIBSVM library as learning algorithm and SVM classification.

Parallel to the development of a graphical interface and the computer code, we tested the different options with the SVM parameter, ranging from the choice of a kernel for the internal operation of data (and obtain the support vectors) to values of margins and deviations.

With computer support and properly parameterized algorithms is tested with and without noise and assessed the strengths and weaknesses of the formulation. The most important and ultimate goal of a theoretical project are the conclusions even the empirical results are not as desired, it always open new means of research.

In this case the simulation results were positive and they draw conclusions in Chapter 5 as well as areas for improvement of existing programming and new approaches that could improve the robustness and reliability.

Completing the project we have the annexes. It is the space to display the numerical results tabulated and guidelines for the use of Matlab functions, both libraries and code algorithms developed exclusively for this proyect.

CAPITULO 1. INTRODUCCIÓN

1.1 INTRODUCCIÓN

El aumento del consumo desde la revolución industrial a nuestros días ha forzado una mayor automatización en la fabricación de bienes para poder aumentar el volumen y reducir los costes. Esta automatización ha evolucionado desde los telares de aprovechamiento hidráulico del siglo XVIII a las complejas fábricas basadas en robótica industrial de nuestros días.

Uno de los elementos empleados en la fabricación son los AGV's (Automatic Guided Vehicle), vehículos que se mueven de manera automática y transportan materiales por rutas predeterminadas de manera ininterrumpida. Estos vehículos llevan sensores de proximidad para detectar obstáculos o cualquier elemento ya sea para no chocar o para interactuar con él.

El presente proyecto abordará el problema del reconocimiento de objetos desde el punto de vista de un robot móvil empleando la técnica de las máquinas de vectores soporte (en adelante SVM, support vector machines). Nuestro robot se moverá por un pasillo de la Universidad Carlos III de Madrid debiendo reconocer puertas, esquinas, rincones, pasillos…

Actualmente el problema del reconocimiento de objetos se puede resolver con otras técnicos basadas, por ejemplo, en redes neuronales. Las SVM son una alternativa que en muchos campos se han mostrado superiores, como en el reconocimiento de datos biométricos (voz, reconocimiento de retina, facial…) por lo que resulta interesante comprobar su potencial en el reconocimiento de modelos en dos dimensiones.

El proyecto se enfoca como un modelo informático en el que probar la validez de las SVM para el estudio de nuestro y después poder extrapolarlo a un robot real.

En el segundo capítulo se hará una introducción de lo que es el aprendizaje supervisado y una descripción de distintos algoritmos para posteriormente centrarnos en las máquinas de vectores soporte. Se analizará su desarrollo teórico y los distintos tipos.

En el tercer capítulo se partirá de un modelo sencillo (diferencia pasillo / hall) para calibrar los parámetros de funcionamiento de nuestro software de SVM y poder asentar los conceptos que se ampliarán en el cuarto capítulo, en el que simularemos suficientes situaciones con distintos niveles de ruido y obstáculos.

En el cuarto capítulo se desarrolla la aplicación en Matlab (1), los modelos creados y los resultados obtenidos de las simulaciones para decidir si el modelo empleado de SVM cumple con los requerimientos de nuestra situación.

En el quinto y último capítulo se exponen las conclusiones y las posibles mejoras que se puedan implementar en futuras versiones.

1.2 OBJETIVOS

- Desarrollar un prototipo en Matlab que permita reconocer y clasificar espacios con SVM a partir de los datos del entorno recogidos por un sensor.
- Estudiar las ventajas e inconvenientes del empleo de SVM para el reconocimiento y clasificación espacial en dos dimensiones.
- Evaluar si resulta aplicable el método de las SVM a nuestro caso de estudio y extraer conclusiones y posibles vías de trabajo futuro.

1.3 METODOLOGÍA

Se empleará como entorno de programación Matlab ya que en él existen toolboxes (funciones y bloque de código desarrolladas por otras personas) de SVM que no están disponibles en otros lenguajes de programación. La librería empleada de SVM será Libsvm (2), desarrollada por Chih-Chung Chang and Chih-Jen Lin. Asimismo se ha aprovechado una función sensor programada en Matlab por el departamento de Señales y Sistemas de la Universidad Carlos III.

Se probará la validez del sistema de reconocimiento realizando modelos sencillos de pasillos y zonas de escalera/halls (capítulo 3), como paso previo a la construcción de una interfaz gráfica que abarcará toda la planta del edificio de la universidad (capítulo 4).

Se han creado funciones que enlazan los datos del sensor con la librería Libsvm. En primer lugar se desarrollará una aplicación de entrenamiento con el objetivo de obtener los vectores que caractericen a los distintos modelos de las zonas estudiadas. A continuación se programará una segunda aplicación para poder llevar a cabo las simulaciones necesarias para cumplir con los objetivos del proyecto.

Finalmente, con los resultados del ensayo, se extraerán conclusiones y futuras mejoras.

1.4 REFERENCIAS

Para el desarrollo del proyecto se han empleado libros de apoyo, artículos y referencias de páginas web. Dicha información bibliográfica está enumerada en el capítulo 6 y citada a lo largo del proyecto.

CAPITULO 2. I.A. Y SISTEMAS DE RECONOCIMIENTO AUTOMÁTICO

2.1 INTELIGENCIA ARTIFICIAL

Definida en 1956 por el informático del MIT John McCarthy como 'la ciencia e ingeniería de hacer máquinas inteligentes, especialmente programas de cómputo inteligente'. Se puede simplificar diciendo que la inteligencia artificial (3) (4) son las inteligencias no naturales en especímenes no vivos.

Estas inteligencias no vivas (de manufactura humana) son capaces de pensar, actuar y evaluar problemas para alcanzar un objetivo basándose en principios de optimización. Sería más correcto decir que presentan racionalidad artificial y no inteligencia, pero el segundo concepto es más comercial y debido a su uso está ya implantado en el vocabulario actual.

Dentro del ámbito de la inteligencia artificial podemos distinguir distintos tipos de conocimientos y de sistemas de representación de dichos conocimientos, los cuales pueden ser aprendidos por la máquina o introducidos en su memoria por un agente experto. Los datos percibidos por la I.A. (5) también serán variados pudiendo ser adquiridos por sensores físicos o mecánicos, pulsos eléctricos… y sus acciones de salida podrán ser tanto informáticas, en forma de bits en un entorno software, como mecánica: accionamientos, desplazamientos…

Podemos dividir la inteligencia artificial en cuatro categorías en función de su forma de pensar y actuar:

- Sistemas que piensan como humanos: tratan de imitar el pensamiento humano. Un ejemplo de ello serían las redes neuronales artificiales. Serían aplicables a actividades relacionadas con formas de pensar humanas como la toma de decisiones, resolución de problemas…
- Sistemas que actúan como humanos: tratan de actuar como humanos imitando su comportamiento. La robótica sería un ejemplo de este tipo de sistemas que buscan mejorar la realización de tareas repetitivas.
- Sistemas que piensan racionalmente: tratan de imitar el pensamiento lógico racional humano, emulando el racionamiento de un experto en un tema concreto. Dentro de esta familia tenemos los sistemas expertos, con los que se busca mejorar en calidad y rapidez en las respuestas para así mejorar la

productividad del experto. Los campos de aplicación más comunes son cálculos, árboles de decisión…

- Sistemas que actúan racionalmente: tratan de emular el comportamiento humano en forma racional. Un ejemplo serían los agentes inteligentes, que son entidades capaces de percibir su entorno, procesar sus percepciones en forma de datos de entrada y actuar de manera racional en dicho entorno.

Las categorías expuestas de inteligencia artificial se dividen en dos escuelas de pensamiento (6), la inteligencia artificial convencional y la inteligencia computacional, que es sobre la que se desarrollan los algoritmos de máquinas de soporte vectorial que se emplearán en el presente proyecto.

2.1.1 INTELIGENCIA ARTIFICIAL CONVENCIONAL

Esta rama de la inteligencia artificial se denomina también IA simbólico-deductiva. Se basa en el análisis estadístico y formal del comportamiento humano. Estos sistemas no implican aprendizaje. Ejemplos de esta escuela de pensamiento serían los siguientes.

Razonamiento basado en casos:

Es un proceso de resolución de problemas que se basa en las soluciones de problemas anteriores, razona empleando analogías. Este método es empleado tanto en máquinas como en humanos, ya que en el día a día el ser humano resuelve situaciones basándose en su experiencia previa.

En este procedimiento es importante tener en cuenta en que parte del problema está el sistema, el lenguaje a emplear, el número, tipo y cantidad de ejemplos empleados, la realimentación del sistema… En función de estas características se particulariza el racionamiento y tendríamos distintos tipos:

- Razonamiento basado en ejemplos. En este caso tenemos un número de ejemplos en la memoria y el problema al resolver uno nuevo estriba en saber si es parte de uno ya existente o si nos enfrentamos a un nuevo ejemplo no clasificado con anterioridad, que deberemos incluir en nuestra base de datos.
- Razonamiento basado en instancias. Mejora al anterior intentando compensar la dificultad de clasificar los nuevos ejemplos empleando un gran número de instancias. Dichas instancias son vectores con los que dirigimos el razonamiento hacia los ejemplos que ya tenemos categorizados.
- Razonamiento basado en memoria. En memoria tenemos una gran colección de ejemplos, por lo que el objetivo es tenerla organizada correctamente. La comparación con la base de datos y el guardado de nuevos ejemplos se basa exclusivamente en criterios sintácticos y no en el significado de los datos.

- Razonamiento basado en casos. A pesar de que se emplea el nombre de manera genérica, también designa un razonamiento más concreto. Un caso tendría mayor entidad que un ejemplo, posee mayor cantidad y riqueza de información por lo que su organización en memoria debe optimizarse para que el acceso a dicha información sea rápido y fluido.
- Razonamiento basado en analogías. Es muy similar al basado en casos, en cuanto a la riqueza de la información, pero en este caso las analogías son más amplias y pueden enfocarse a dominios diferentes. Se busca relacionar experiencias distintas pero con semejancias entre ellas.
-

Redes bayesianas

Son modelos probabilísticos que relacionan variables aleatorias mediante un grafo dirigido que indica influencia casual En este modelo, los nodos representan variables y los arcos las dependencias condicionales entre ellas. Se basa en el teorema de Bayes de probabilidad condicionada y son muy útiles para estimar probabilidades de nuevos ejemplos y para modelos de minería de datos.

Sistemas expertos

Como se comentó con anterioridad, un sistema experto es aquel que simula el comportamiento de un experto en un tema concreto, ya sea porque fue programado por uno o porque el sistema es usado por un experto.

Son aplicaciones informáticas que necesitan un gran conocimiento sobre el tema a tratar y para que sean efectivos deben reunir las siguientes capacidades:

- Explicar sus razonamientos: al usuario de manera completa y sencilla a partir de los hechos en los que se basan
- Adquisición de nuevos conocimientos: que modifiquen a los anteriores que tienen en la base de datos si fuera necesario, no para corregir al experto que los programó si no para mejorar la eficacia y rapidez en el desarrollo de sus tareas.

Para llevar a cabo sus funciones, los sistemas expertos tienen características comunes que son:

- Base de datos o conocimientos: conformada por los datos del experto
- Base de hechos: memoria de hechos y soluciones extraídas de los problemas resueltos y analizados por el sistema
- Motor de inferencia: que modela el pensamiento humano
- Módulos de justificación: explica el razonamiento seguido por el sistema experto para llegar un resultado
- Interfaz: en lenguaje natural para facilitar la interacción con el usuario.

Los sistemas expertos se apoyan en los dos tipos de métodos de inteligencia artificial convencional antes descritos. Tenemos sistemas expertos basados en casos (llamados CBR, *Case Based Reasoning*) donde la solución de un nuevo problema se basa en un

problema similar planteado con anterioridad. También existen los basados en redes bayesianas, aplicando el teorema de Bayes y estadística.

Otro modelo de sistema experto es el basado en reglas heurísticas apoyadas en lógica difusa. Estos trabajan mediante la aplicación de reglas, comparación de los resultados y aplicación de reglas modificadas en función de la situación analizada. También pueden emplear lógica dirigida, empezando por ejemplo en una situación con una evidencia inicial y dirigiéndose hacia una solución, o realizando hipótesis acerca de las posibles soluciones y volviendo atrás para encontrar una evidencia que apoye las primeras hipótesis tomadas.

2.1.2 INTELIGENCIA COMPUTACIONAL

La inteligencia computacional es la rama de la I.A. conocida como subsimbólica-inductiva que implica aprendizaje interactivo basado en datos empíricos. Se centra en el estudio de mecanismos adaptativos que permitan al sistema comportarse de manera inteligente sin emplear algoritmos heurísticos, como hace la inteligencia artificial convencional.

Combina elementos de aprendizaje, evolución, adaptación y lógica difusa, sin dejar de lado elementos estadísticos, pero siempre en segundo plano, que aportan un soporte complementario para los programas y algoritmos.

Dentro de esta escuela de pensamiento encontraremos métodos como las redes neuronales, computación evolutiva, sistemas difusos y otros métodos de aprendizaje automático como las *máquinas de soporte vectorial*.

Ya definida la inteligencia artificial y sus clases, se desarrollarán en los siguientes epígrafes los sistemas de aprendizaje automático y las máquinas de soporte vectorial, que son el objetivo del presente proyecto.

2.2 SISTEMAS DE RECONOCIMIENTO AUTOMÁTICO

Un sistema de identificación o reconocimiento automático se puede definir como una técnica o conjunto de ellas que clasifican o distinguen un elemento después de haber analizado ciertas características.

Hay muchas familias de sistemas de reconocimiento automático: para superficies, imágenes, lenguajes, datos biométricos, datos atmosféricos… En lo relativo a este proyecto se aplicará al reconocimiento de superficies y áreas en dos dimensiones mediante el análisis de los elementos característicos de las distintas habitaciones y espacios.

2.2.1 FUNCIONAMIENTO DE UN SISTEMA DE RECONOCIMIENTO AUTOMÁTICO

El sistema de reconocimiento clasificará los espacios del mapa de la universidad en función de los patrones que ha aprendido con anterioridad.

En la figura 1 se muestra el esquema de funcionamiento para este tipo de sistemas.

La línea punteada marca la frontera entre el sistema físico (hardware), y el sistema informático (software y SVM).

El sensor montado en el robot es el encargado de capturar los datos del entorno que son las entradas a nuestro sistema. Estos datos se procesan para adaptarlos a nuestras necesidades (cambios de origen, agrupaciones…) y en la siguiente etapa se identifican las características del muestreo (los diferentes objetos). Teniendo un listado de objetos los comparamos con nuestra base de datos de patrones ya aprendidos y después de normalizar las coincidencias, se toma una decisión.

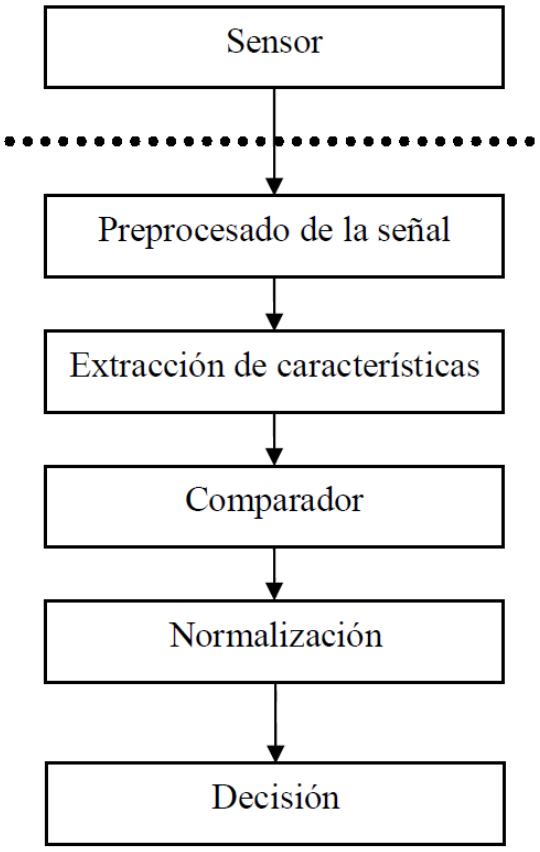


Figura 1 Esquema de un sistema de reconocimiento automático

2.3 APRENDIZAJE

2.3.1 INTRODUCCIÓN

El aprendizaje es una rama de la inteligencia artificial que busca desarrollar técnicas que permitan a una máquina o computadora aprender. El hecho de aprender está basado en el análisis de datos y la extracción de conclusiones lo que liga esta disciplina con la estadística (tanto en su rama descriptiva como inferencial).

De forma práctica consiste en el desarrollo de programas informáticos que puedan generalizar comportamientos partiendo de información que le hayamos suministrado. Dichos programas se basan en algoritmos de aprendizaje que sirven para mejorar la realización de alguna tarea a través de la experiencia, con lo cual van 'aprendiendo' hasta ser capaces de inducir una función capaz de resolver el problema al que se enfrentan.

Los distintos tipos de algoritmos se pueden agrupar en función de su salida. A continuación se enumeran una serie de ellos:

Aprendizaje supervisado

Estos algoritmos presentan una correspondencia entre la salida y las entradas del sistema. El punto de partida son los ejemplos etiquetados en la etapa de aprendizaje, con los que se crean modelos que son la base contra la que trabaja el algoritmo.

Un ejemplo serían los problemas de clasificación, donde se etiquetan vectores utilizando una serie de categorías o clases.

Aprendizaje no supervisado

En este caso solo existen datos de entrada por lo que no se tiene información a priori sobre las categorías. Al no existir modelos el sistema tiene que ser capaz de reconocer patrones con los que etiquetar los datos suministrados al algoritmo.

Aprendizaje semi-supervisado

Es una mezcla de las dos categorías anteriores. Se combinan los algoritmos de aprendizaje supervisado y no supervisado para poder clasificar adecuadamente.

Aprendizaje por refuerzo

El algoritmo o función aprende por el método de ensayo-error. Recibe datos de su entorno como respuesta a sus acciones, lo que supone una retroalimentación o feedback que se convierte en su información de entrada.

Transducción

Sigue el mismo procedimiento que el aprendizaje supervisado, pero en vez de construir una función, se trata de predecir la clase de salida en función de los ejemplos de entrada y sus respectivas clases.

Aprendizaje multi-tarea

Es un método de aprendizaje que para resolver problemas emplea conocimientos que ha aprendido previamente que sean similares a dichos problemas.

De esta lista de sistemas de aprendizaje en el presente proyecto vamos a profundizar en el aprendizaje supervisado ya que el método que emplearemos para reconocer localizaciones serán los algoritmos basados en máquinas de soporte vectorial (SVM).

En la figura 2 se muestra una clasificación de los tipos de aprendizaje inductivo. Este tipo de aprendizaje se basa en que el sistema pueda conseguir los conocimientos necesarios para realizar la tarea para la cual está diseñado a partir de ejemplos reales. En el lado opuesto estaría el aprendizaje deductivo, en el que se transfieren los conocimientos que un experto humano posee al sistema.

En la práctica, no existen sistemas inductivos puros, ya que los aspectos generales del problema suelen ser aportados deductivamente, mientras que la generalización de éste se desarrolla inductivamente, por lo que se debería hablar de aprendizaje deductivo-inductivo.

De acuerdo con los expuesto anteriormente, dentro del aprendizaje inductivo y en función de la tarea que se necesite resolver se pueden emplear algoritmos predictivos (aprendizaje supervisado) y descriptivos (no supervisado).

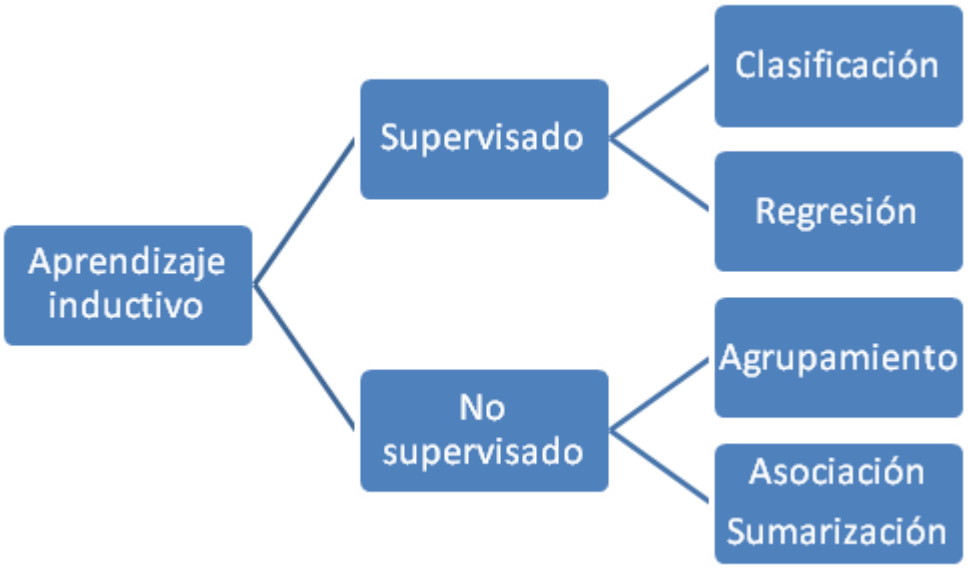


Figura 2 Clasificación de métodos de aprendizaje inductivo

Los algoritmos que siguen modelos predictivos son los llamados de aprendizaje supervisado. Para cada muestra evaluada por la función se conoce la salida, denominada etiqueta, que sirve para evaluar la capacidad de acierto de la predicción realizada. Si la etiqueta es discreta, el modelo será de clasificación, mientras que si es continua se realiza una tarea de regresión.

Las máquinas de soporte vectorial son algoritmos predictivos y existen ambos tipos, tanto SVM de clasificación como SVM de regresión. La toolbox Libsvm permite trabajar con los dos algoritmos, pero en nuestro caso, dado que emplearemos etiquetas binarias (0 o 1) solo se utilizará el modelo de clasificación.

Los algoritmos que siguen modelos descriptivos son conocidos por aprendizaje no supervisado, ya que no tienen un atributo salida. Los datos de entrada suelen ser variables aleatorias sobre las que se construye un modelo de densidad. Se dividen generalmente en: agrupamiento, los datos se agrupan según su similaridad; sumarización, que busca hallar una descripción para el conjunto de datos y asociación, cuyo objetivo es encontrar patrones de asociación entre los atributos de los datos de entrada.

2.3.2 APRENDIZAJE SUPERVISADO

Como se comentó anteriormente el aprendizaje supervisado es una técnica que permite deducir una función a partir de datos de entrenamiento. Estos datos normalmente consisten en pares de objetos (usualmente vectores), con una componente que son los datos de entrada y el otro el resultado deseado. Si el aprendizaje es de tipo regresión, la salida de la función puede ser un valor numérico, mientras que si es de tipo clasificación será una etiqueta de la clase resultado.

La función debe ser capaz de predecir el valor correspondiente a cualquier objeto de entrada después de haber sido entrenada con varios ejemplos.

Los pasos típicos a seguir para resolver un problema con aprendizaje supervisado suelen ser:

1. Determinar los ejemplos de entrenamiento. Decidir el tipo de dato que se va a emplear, ej. En números, si emplear el número entero o dividir cifra a cifra.
2. Obtener un conjunto de datos de entrenamiento, ya sean manualmente, recibidos de un sensor, imaginarios, ejemplos reales…
3. Determinar el formato de los datos de entrada. Partiendo de los ejemplos de entrenamiento, sacar las características a estudiar y desarrollar una función de ingreso de dichas características a nuestro entorno de programación.
4. Determinar la función que resolverá el problema. Decidir si se emplean árboles de decisión, redes neuronales, máquinas de vectores soporte…

5. Finalizar el diseño. Ajustar los parámetros del algoritmo y validar los datos de entrenamiento.

2.3.3 APRENDIZAJE ACTIVO

El aprendizaje activo es una variedad del aprendizaje supervisado que se emplea en situaciones en las que hay muchos datos sin etiquetar, mientras que resulta caro obtener datos etiquetados. En esta situación el algoritmo consulta de manera activa al experto humano para las etiquetas. Dato que el experto elige las etiquetas el número de ejemplos necesarios es mucho menor que en aprendizaje supervisado normal.

2.3.4 SISTEMAS DE APRENDIZAJE SUPERVISADO

Hay muchos ejemplos de sistemas de aprendizaje supervisado:

- Redes neuronales (perceptrón)
- Máquinas de vectores soporte
- Clasificador bayesiano ingenuo
- Modelo de los k-vecinos más próximos
- Árboles de decisión
- Funciones de base radial

Antes de entrar en detalle con las máquinas de soporte vectorial se explicarán las bases de funcionamiento de otros métodos de aprendizaje supervisado para tener una visión más amplia del tema.

2.3.4.1 REDES NEURONALES

Denominadas abreviadamente RNA (7) se inspiran en el funcionamiento del sistema nervioso de los seres vivos. Se trata de un sistema de neuronas interconectadas en red que colaboran para obtener una salida.

El objetivo de imitar las conductas de los sistemas neuronales biológicos mediante modelos matemáticos es que las máquinas sean capaces de dar respuestas similares a las que es capaz de dar el cerebro.

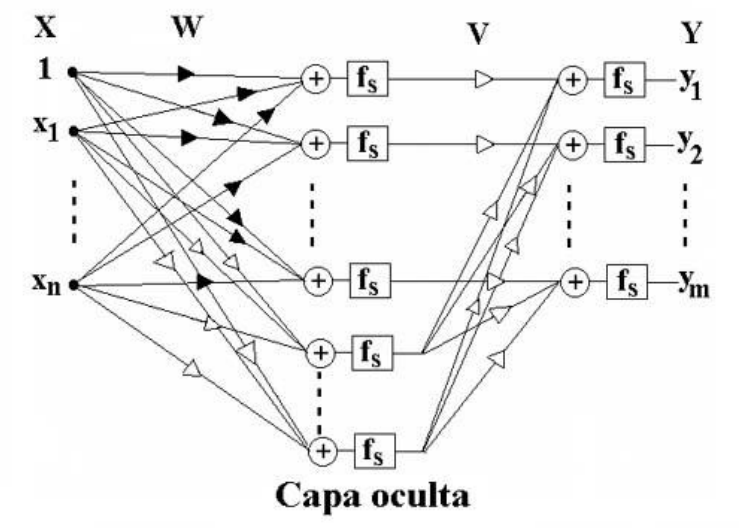


Figura 3 Ejemplo de red neuronal (7)

La unidad elemental de las redes son las neuronas. Cada una de ellas recibe una serie de entradas a través de todas sus conexiones y dan una sola salida que viene dada por las siguientes funciones:

- Función de excitación o de propagación. Consiste en el sumatorio de cada una de las entradas a la neurona multiplicadas por el valor neto o peso de su interconexión. Si el valor es positivo se llama excitatoria, mientras que si es negativo se denomina inhibitoria.
- Función de activación. Esta función modifica al anterior. No siempre está presente y al no estarlo la salida es la misma que de la función de propagación.
- Función de transferencia. Se aplica al valor resultante de la función de activación para acotar la salida. Por ejemplo entre 0 y 1 empleando una función sigmoidal.

El funcionamiento de una red neuronal se divide en dos etapas, la etapa o fase es la de activación y la segunda la de aprendizaje. Estas etapas son secuenciales, empieza por la primera y a continuación realiza la segunda ya que su método de aprendizaje es iterativo por lo que después del aprendizaje emplea la función de activación para retroalimentar sus errores y calibrar el modelo (8).

La etapa de activación es en la que un valor pasa a través de la red neuronal y como salida obtenemos un valor que será la clasificación o estimación de los inputs. Los pasos que se llevan a cabo en esta etapa son:

- Procesar los inputs o datos de entrada
- Multiplicar cada dato de entrada por una ponderación (peso específico o sinapsis) y crear un valor resumen que es el sumatorio de dichos datos ponderados.
- Se le aplica la función de activación al valor resumen

- Se obtiene un valor de salida que es enviado a la siguiente neurona de la red como dato de entrada.

En la etapa de aprendizaje la red modifica sus pesos específicos de ponderación en función de una información de entrada. Pueden darse tres tipos de cambios en las conexiones entre neuronas de la red:

- Creación: se crea una sinapsis entre dos neuronas con un valor distinto de 0
- Destrucción: la sinapsis desaparece, esto es, pasa a valer 0
- Modificación: el peso específico cambia de valor

Entre los distintos tipos de algoritmos de aprendizaje, se suele emplear el de la retropropagación en los perceptrones multicapa. Es un tipo de aprendizaje por corrección del error de la predicción ajustando las ponderaciones en función de los valores de salida de la red neuronal y los que realmente deseamos. Esta corrección se hace desde las últimas neuronas de la red (las más cercanas a la salida) hacia atrás.

Algunas del as ventajas del empleo de redes neuronales son:

- Al crear una interrelación compleja en la red, entre las diferentes neuronas y los distintos valores de las sinapsis, presenta una alta resistencia a la información incompleta y al ruido
- Las redes responden de manera correcta a entradas no conocidas, realizando eficientemente el proceso de generalización en el análisis de los datos.

Una vez introducidas las redes neuronales, se desarrollan en los siguientes epígrafes dos modelos de redes, el perceptrón y el perceptrón multicapa.

Perceptrón

El perceptrón (9) es un modelo de red neuronal simple creado por Frank Rosenblatt que puede emplearse como neurona artificial sola o dentro de otro perceptrón más grande. Este modelo permite representar mediante una salida la presencia de uno o varios fenómenos a su entrada.

Al igual que una neurona sola en un sistema biológico carecería de sentido, el empleo del perceptrón cobra importancia al emplearlo en redes más grandes, donde la salida de cada neurona/perceptrón alimenta a otras neuronas. De esta forma también tenemos que cada unidad neuronal puede tener varias entradas como se muestra en la figura 4, con lo que el elemento podrá discriminar entre sucesos de entradas variables y dar una salida que sea resultado de la interacción de varias entradas.

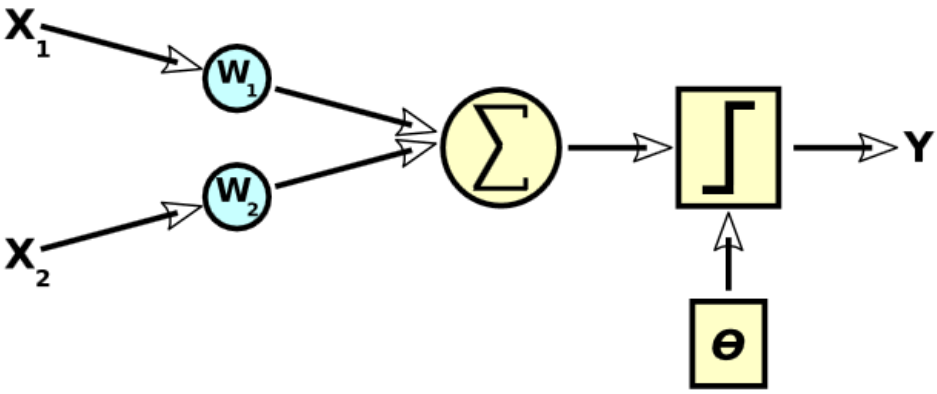


Figura 4 Representación de un perceptrón (10)

Perceptrón multicapa

El perceptrón multicapa (11) es una RNA (red neuronal artificial) formada por perceptrones simples conectados entre sí en forma de múltiples capas, lo que lo capacita para resolver problemas que no son separables linealmente.

Podemos diferenciar entre perceptrones totalmente conectados, en los que la salida de una neurona es entrada de todas las neuronas de la capa sucesiva y perceptrones localmente conectados en los que la salida de una neurona no tiene porqué estar conectada a todas las neuronas de la siguiente capa.

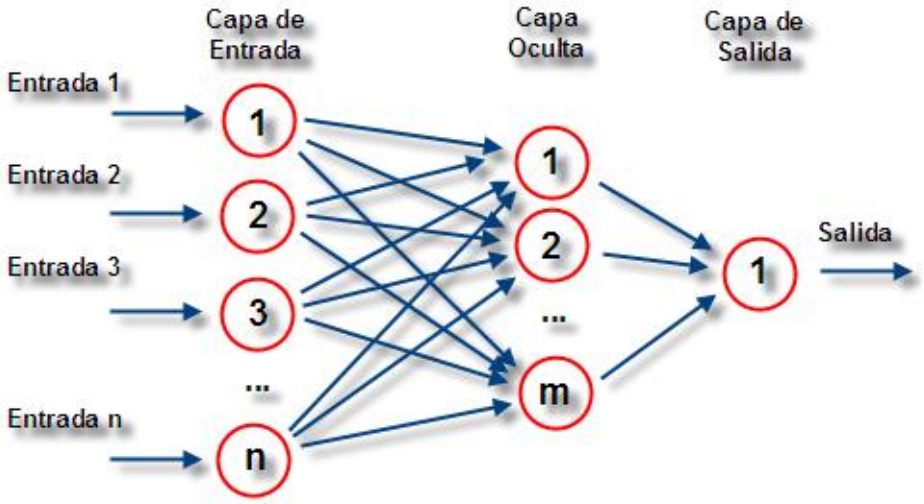


Figura 5 Representación de un perceptrón multicapa

En la figura 5 se muestra un ejemplo de representación de un perceptrón multicapa. Dichas capas pueden clasificarse en tres tipos (12):

- Capa de entrada: en ella están las neuronas por las que entra información a la red. En esta no se procesa información.
- Capas ocultas: esta capa está formada por neuronas que tienen entradas provenientes de otras neuronas y cuyas salidas también llegan a otras neuronas.
- Capa de salida: los valores de salida de las neuronas pertenecientes a esta capa son las salidas de la red neuronal.

El algoritmo de aprendizaje de la retropropagación comentado en el apartado 2.3.4.1 es aplicable a esta configuración de red.

2.3.4.2 ÁRBOLES DE DECISIÓN

El método de los árboles de decisión es un modelo que partiendo de una base de datos y creando diagramas de construcción lógica sirve para representar y categorizar una serie de sucesos que ocurren de forma sucesiva para la resolución del problema.

Un árbol de decisión posee unas entradas y a partir de ellas devuelve una respuesta, que es un conjunto de decisiones tomadas en función de las entradas. Las entradas pueden ser discretas o continuas aunque lo normal es emplear, por simplicidad, valores discretos (numéricos). Como se explicó en el apartado 2.2.1, al emplear valores discretos como entrada del árbol, se trataría de una aplicación de clasificación mientras que si utilizamos valores continuos se denominaría de regresión.

Los árboles de decisión están formados de distintos elementos o nodos:

- Nodos internos: contienen un test sobre algún valor de una de las propiedades. Las decisiones que se tomen en estos nodos no son probabilísticas. Su representación es mediante un cuadrado.
- Nodos de probabilidad: marcan la probabilidad de algún evento aleatorio que tenga relación con el problema. Se representa con un círculo.
- Nodos hoja: marcan el valor que devolverá la resolución del problema.

La unión de los nodos se realiza mediante ramas, que son las que brindan los posibles caminos de acuerdo con las decisiones que se tomen en cada nodo.

Para resolver el problema con el árbol de decisión hay que calcular el valor de cada uno de los resultados, evaluando las consecuencias inciertas en los nodos de probabilidad multiplicando el valor en esos nodos por la probabilidad de que ocurran los sucesos. Cuando se tiene todos los resultados se puede tomar la decisión más beneficiosa.

Las ventajas de este método son:

- Plantear el problema de manera clara con todas sus opciones
- Se basa es un esquema que cuantifica el coste de cada resultado y la probabilidad de que éste suceda.
- Se analizan todas las consecuencias de las decisiones tomadas.

2.3.4.3 MÉTODO DE K-NN

El método k-nn (13) (K nearest neighbors, *Fix yHodges, 1951*) es un método de aprendizaje supervisado de tipo clasificación que sirve para estimar la función de densidad $F(x/C_j)$. A partir de la información que se generó a través de los datos de entrenamiento, estima la probabilidad de que un elemento x pertenezca a la clase C_j, no haciendo ninguna suposición acerca de la distribución de las variables predictoras.

Los ejemplos de entrenamiento son vectores de un espacio multidimensional, teniendo *p* atributos y considerando *q* clases para la clasificación.

Los valores de los atributos se representarían por el vector *p*-dimensional:

$$x_i = (x_{1i}, x_{2i}, \dots, x_{pi})$$

En función de los datos de entrenamiento, el espacio se divide en tantas localizaciones como clases haya. La clase *C* de un punto se designa cuando ésta sea la clase más frecuente entre los ejemplos de entrenamiento más cercanos al punto en cuestión. Para localizarlos se suele emplear la distancia euclídea.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p} (x_{ir} - x_{jr})^2}$$

Durante la fase de entrenamiento se almacenan vectores y sus respectivas clases para tener una base de datos sobre la que trabajar en la fase de clasificación. En ésta, la entrada es un vector de clase desconocida. Partiendo de dicho vector del espacio característico del problema se calculan las distancias entre él y los vectores de entrenamiento almacenados. De los k vectores más cercanos se comprueba cual es la clase que más se repite y ésta es la que se asigna al nuevo vector.

En éste método se supone que los vectores más próximos son los que dan una mejor clasificación, entendiendo que por proximidad se puede conjeturar que pertenecen a la misma clase, pero el problema es que esto se hace considerando todos los atributos cuando algunos podrían ser irrelevantes y distorsionar la clasificación por no estar ponderados.

Por lo que versiones más exactas del algoritmo consisten en identificar los atributos más relevantes y ponderarlos de forma que podamos acotar correctamente la elección de clase para los nuevos ejemplos de clasificación.

2.4 MÁQUINAS DE SOPORTE VECTORIAL

Las máquinas de soporte vectorial (14) surgen como un método de clasificación basado en la teoría de Vapnik y de su equipo de AT&T de minimización de riesgo estructural. Teniendo un conjunto de datos de muestra o ejemplos de entrenamiento etiquetamos sus clases y entrenamos una SVM construyendo un modelo que será capaz de predecir la clase de los nuevos datos que le introduzcamos. Intuitivamente la SVM representa en un eje de coordenadas los vectores de entrenamiento, separando las clases presentes en los ejemplos por un espacio lo mayor posible, cuando introducimos nuevos datos se colocan sobre el mismo eje y en función de su proximidad a uno de los grupos antes separados son clasificados en una u otra clase.

Actualmente tienen muchas aplicaciones debido a sus prestaciones y a su versatilidad. Las SVM (15) se han empleado con éxito en campos como el reconocimiento de textos o escritura, recuperación de información o la clasificación de imágenes.

El primer paso, antes de poder clasificar, es realizar una etapa de aprendizaje. Consiste en encontrar el hiperplano $h(x) = 0$ que mejor separe un conjunto de datos $X \in \Re^d$ según la clase $Y \in \{-1,1\}$ a la que pertenecen. Dicho hiperplano es el que maximiza la distancia al punto más próximo de cada clase, por lo tanto, estará a la misma distancia de los ejemplos más cercanos de cada categoría.

Según Vapnik, el separador lineal que maximiza el margen (el doble de la distancia al punto más próximo en cada clase) es el que da la mayor capacidad de distinguir características comunes de los datos de cada clase que permitan clasificar datos que no sean los del conjunto de entrenamiento. Para hallar dicho separador, se resuelve un problema de optimización empleando técnicas de programación cuadrática.

A los datos empleados para hallar el hiperplano (frontera de decisión), se los conoce como vectores de aprendizaje o de entrenamiento, al igual que en otros algoritmos de aprendizaje supervisado. Estos vectores son los que permiten crear los modelos con los que trabaja la SVM para clasificar los nuevos datos que le introduzcamos.

A partir de unos datos de entrada x_i, las SVM nos proporcionarán su clase según la regla de clasificación $f(x_i) = signo(h(x_i))$.

En la figura 6 se representan datos de dos clases (cuadrados azules y círculos verdes) separados por el hiperplano que maximiza la distancia entre ellos. Esta distancia es la marcada como margen, que es máxima para el hiperplano obtenido en este caso, cualquier otro hiperplano presentaría un margen de separación de clases menor y por lo tanto sería menos adecuado.

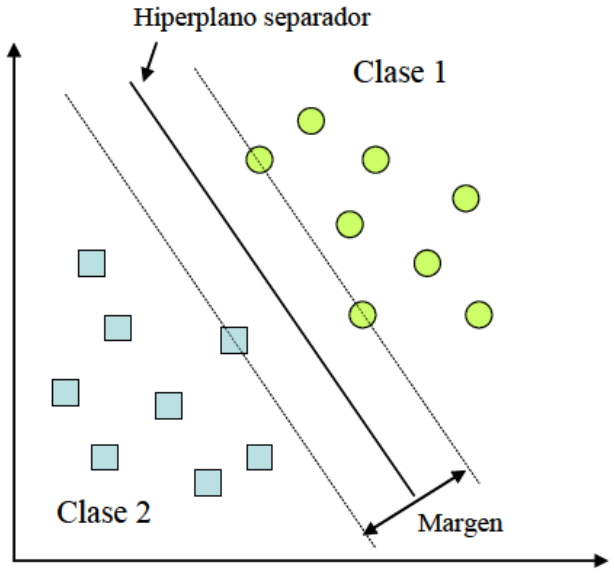


Figura 6 Separación de datos mediante SVM

Tras el aprendizaje se comprueba el error cometido tomando una nueva muestra de datos (*conjunto de test o validación*) y comparando la salida que obtenemos con su clase real. De cada muestra de datos se suele tomar un 75% como vectores de aprendizaje y el 25% de los vectores restantes se utilizan para poder contrastar la fiabilidad de la frontera de decisión obtenida.

La calibración del modelo es fundamental para que dichos modelos sean sólidos y recreen con fiabilidad la situación deseada. En caso de que no consigamos el resultado deseado se deberán aumentar los datos de entrenamiento hasta obtengamos una clasificación correcta.

El reto a la hora de entrenar una SVM es hacerlo con un grupo mínimo de datos de entrenamiento, ya que a más datos mayor es el coste del aprendizaje. Conseguir ratios altos de fiabilidad/coste debe ser el objetivo buscado.

2.4.1 MÁQUINAS DE SOPORTE VECTORIAL PARA CLASIFICACIÓN BINARIA

En los procesos de clasificación binaria solo existen 2 clases: una es considerada como positiva (y = 1) y la otra como negativa (y = -1). El valor de la etiqueta no es importante, ya que dependerá del material bibliográfico seguido por el autor o de las limitaciones en el lenguaje de programación. Podrá tomar parejas de valores como positiva:1 negativa:-1, positiva:1 negativa:0, positiva:0 negativa:-1 …

La situación más cómoda para resolver con SVM, en cuanto al tipo de datos que tengamos, es que dichos datos sean linealmente separables pero las entradas es muy posible que sean no separables linealmente o que exista un cierto nivel de ruido en las medidas que las distorsione. En estas situaciones se pueden emplear distintos tipos de SVM que se explicarán en los siguientes epígrafes:

- SVM lineal con margen máximo
- SVM para la clasificación no lineal
- SVM con margen blando.

2.4.1.1 SVM LINEAL CON MARGEN MÁXIMO

Solo se deberían emplear cuando los datos son separables linealmente (16), es decir, se puede usar como frontera de decisión un hiperplano $h(x_i)$ tal que:

$$h(x) = \omega^T x + b = 0$$

donde ω y $x \in \Re^d$, siendo d la dimensión del espacio de entrada.

La resolución para dicho caso sería suponer que se tiene un conjunto de n datos separables linealmente $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ donde $x_1 \in \Re^d$ e $y_1 \in \{-1,1\}$. Se cumplirá, según el lado en el que se encuentren respecto del hiperplano:

$$\omega^T x_i + b > 0, \text{ para } y_i = 1, i = 1, \dots, n$$

$$\omega^T x_i + b < 0, \text{ para } y_i = -1, i = 1, \dots, n$$

Las ecuaciones definen las dos clases presentes en nuestro problema que al no estar mezcladas, esto es, son linealmente separables, nos permiten hallar con sencillez a nivel matemático el hiperplano que las separa con margen máximo.

Las expresiones antes halladas las podemos reducir a una sola:

$$y_i(\omega^T x_i + b) > 0, \text{ para } i = 1, \dots, n$$

Para resolver el problema, se considera que los vectores soporte (los puntos más cercanos al hiperplano), cumplen:

$$h(x_i) = 1, \text{ para } y_i = 1$$

$$h(x_i) = -1, \text{ para } y_i = -1$$

Los vectores soporte están representados con relleno en color negro en la figura 7. Al ser una aproximación sencilla los vectores soporte son fácilmente identificables a nivel gráfico.

No puede haber datos del conjunto de aprendizaje dentro del margen, por definición del método, por lo que la ecuación $y_i(\omega^T x_i + b) > 0$ queda:

$$y_i(\omega^T x_i + b) \geq 1, i = 1, \dots, n$$

La distancia $dist(h,x)$ de un punto al hiperplano es:

$$dist(h,x) = \frac{|h(x)|}{\|\omega\|}$$

Como los puntos más próximos al hiperplano cumplen $|h(x)| = 1$, su distancia al hiperplano sería:

$$dist(h,x) = \frac{1}{\|\omega\|}$$

Para encontrar los valores de ω y b hay que resolver un problema de optimización que consiste en maximizar la distancia $dist(h,x)$ entre el hiperplano y el punto de entrenamiento más próximo:

Maximizar:

$$\frac{1}{\|\omega\|}$$

Sujeto a:

$$y_i(\omega^T x_i + b) \geq 1, i = 1, \dots, n$$

Que es la condición de que ningún vector de entrenamiento quede dentro del margen que separa a las dos clases.

En la figura 7 queda planteado el problema con sus elementos característicos: las dos clases (idénticas a las de la figura 6), el hiperplano a hallar y el margen máximo.

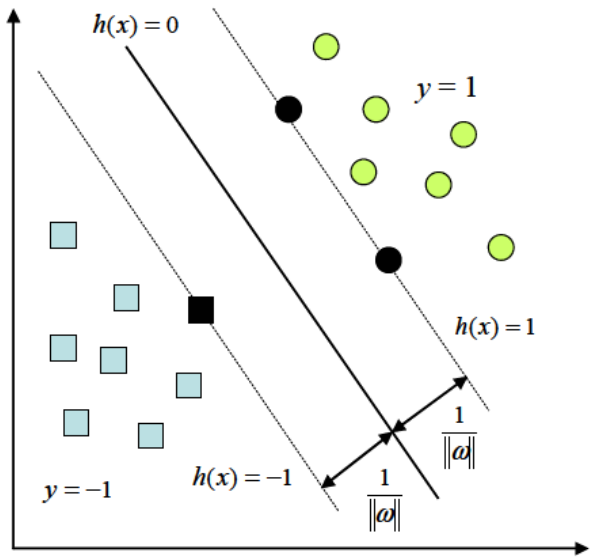


Figura 7 SVM con margen máximo (en negro representados los vectores soporte)

Se puede demostrar que al maximizar $1/\|\omega\|$ se obtiene la misma solución que al minimizar $\|\omega\|^2/2$.

El problema se puede exponer en su formulación dual que es más sencilla de resolver. Como es un problema de programación no lineal se emplean multiplicadores de Lagrange y las condiciones de Karush-Kuhn-Tucker, que en realidad son una generalización de los multiplicadores. Como se comentó, el objetivo será minimizar $\|\omega\|^2/2$ sujeto a $y_i(\omega^T x_i + b) \geq 1, i = 1, \dots, n$

El planteamiento es el siguiente:

$$L(\omega, b, \alpha) = \frac{1}{2}\|\omega\|^2 + \sum_{i=1}^{n} \alpha_i\big(1 - y_i(\omega^T x_i + b)\big)$$

$$\frac{\partial L(\omega, b, \alpha)}{\partial \omega} = 0 \rightarrow \omega = \sum_{i=1}^{n} y_i \alpha_i x_i$$

$$\frac{\partial L(\omega, b, \alpha)}{\partial b} = 0 \rightarrow \sum_{i=1}^{n} y_i \alpha_i = 0$$

$$\alpha_i\big(1 - y_i(\omega^T x_i + b)\big) = 0 \qquad 1 \leq i \leq n$$

$$1 - y_i(\omega^T x_i + b) \leq 0 \qquad 1 \leq i \leq n$$

$$\alpha_i \geq 0 \qquad 1 \leq i \leq n$$

Según las condiciones de Karush-Kuhn-Tucker, en el caso en el que los datos no son vectores soporte, $\alpha_i = 0$

Sustituyendo en la ecuación de Lagrange, se obtiene la función objetivo de la formulación dual:

$$L(\omega, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j {x_i}^T x_j$$

De esta nueva función objetivo debemos obtener los valores de ω y b que definen nuestro hiperplano. Esto se consigue al maximizar:

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j {x_i}^T x_j$$

Sujeto a las condiciones:

$$\sum_{i=1}^{n} \alpha_i \, y_i = 0 \qquad \alpha_i \geq 0 \qquad 1 \leq i \leq n$$

Recordamos a la hora de resolver que los datos que no son vectores soporte tendrán un valor de α_i igual a 0.

Al resolver se obtiene la siguiente solución:

$$\omega = \sum_{i=1}^{n} \alpha_i \, y_i x_i$$

$$h(x) = \omega^T x + b = \sum_{i=1}^{n} y_i \alpha_i {x_i}^T \, x + b$$

$$b = -\frac{1}{2}\left(\max_{y_i=-1} \{\omega^T x_j\} + \min_{y_i=1} \{\omega^T x_j\} \right)$$

Se puede observar que la ecuación del hiperplano solo depende de los vectores soporte ya los puntos restantes cumplen $\alpha = 0$. Esto significa que se llegaría a la misma solución si se volviera a calcular la frontera de decisión únicamente con los vectores soporte. De ahí le viene el nombre al algoritmo, máquinas de soporte vectorial.

2.4.1.2 SVM PARA LA CLASIFICACIÓN NO LINEAL

Es posible que los datos de entrada no sean linealmente separables (17) como se muestra en la figura 8.

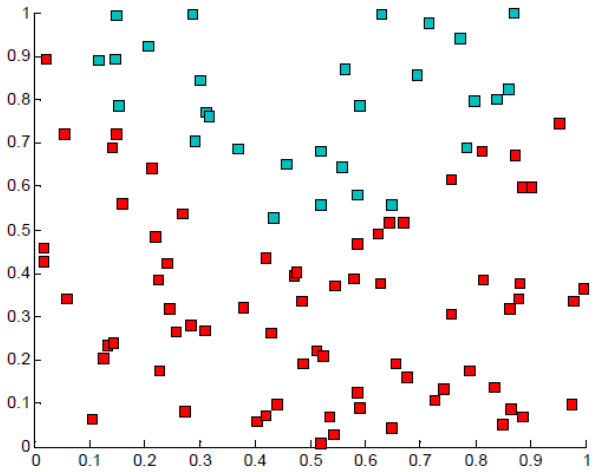


Figura 8 Ejemplo de un conjunto de datos no linealmente separables

Cuando esto sucede, existe la posibilidad de transformar los datos a un espacio ζde mayor dimensión (el espacio de características) en el que los puntos si pueden ser separados por un hiperplano. Para ello, se utiliza una función Φ, tal que:

$$\Phi: \quad \Re^d \rightarrow \zeta$$
$$x \rightarrow \Phi(\mathrm{x})$$

La función Φ, como muestra la figura 9, mueve los datos de entrada que no son linealmente separables a un espacio de mayor dimensión donde si podremos encontrar una hiperplano que los separe.

La frontera de decisión resultante en el espacio de entrada ya no será lineal y vendrá dada por otro tipo de función que puede ser polinómica de grado distinto a 1, gaussiana…

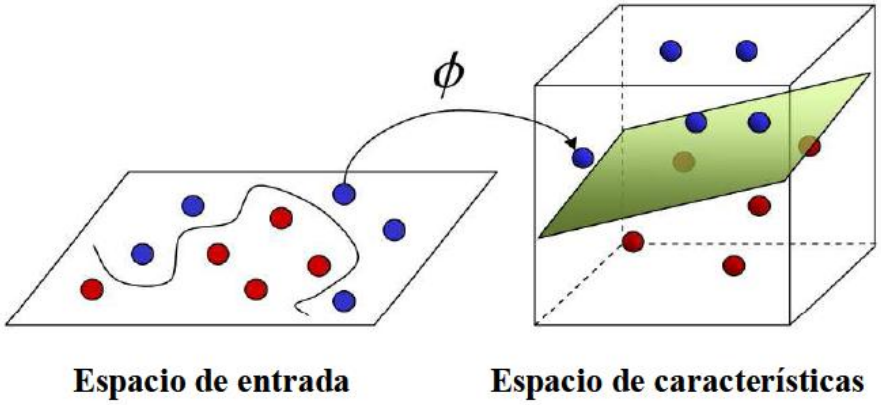


Figura 9 Transformación de los datos de entrada a un espacio de mayor dimensión

Las funciones que se usan para poder realizar esta transformación se llaman funciones núcleo o kernel. Representan el producto vectorial en el espacio de características.

Si tuviéramos una transformación Φ de $\Re^2$ en $\Re^3$: $\Phi(\mathrm{x}_1, \mathrm{x}_2) = \left(\mathrm{x}_1{}^2, \sqrt{2}\mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2{}^2\right)$, el producto escalar $< \Phi(\mathrm{x}), \Phi(\mathrm{x}') >$ en el espacio de características sería:

$$< \Phi(\mathrm{x}), \Phi(\mathrm{x}') >= \left(\mathrm{x}_1{}^2, \sqrt{2}\mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2{}^2\right) \cdot \left(\mathrm{x}_1{}'^2, \sqrt{2}\mathrm{x}_1{}'\mathrm{x}_2{}', \mathrm{x}_2{}'^2\right)^{\mathrm{T}}$$
$$= \left((\mathrm{x}_1, \mathrm{x}_2) \cdot (\mathrm{x}_1{}', \mathrm{x}_2{}')^{\mathrm{T}}\right)^2 = \ < x, \mathrm{x}' >^2$$

Por tanto, el producto escalar $< \Phi(\mathrm{x}), \Phi(\mathrm{x}') >$ en el espacio de características se calcula como $< x, \mathrm{x}' >^2$, es decir, a partir del producto escalar en el espacio de entrada. El kernel asociado al espacio de características es $\mathrm{K}(\mathrm{x}, \mathrm{x}') =< x, \mathrm{x}' >^2$. La función núcleo permite calcular el producto escalar $< \Phi(\mathrm{x}), \Phi(\mathrm{x}') >$ sin tener que calcular la transformación Φ.

En las máquinas de soporte vectorial de margen máximo, la solución era:

$$\omega = \sum_{i=1}^{n} \alpha_i \, y_i x_i$$

$$h(x) = \omega^T x + b = \sum_{i=1}^{n} y_i \alpha_i {x_i}^T x + b$$

$$b = -\frac{1}{2}\left(\max_{y_i=-1} \{\omega^T x_j\} + \min_{y_i=1} \{\omega^T x_j\} \right)$$

Para lograr la frontera de decisión para la SVM no lineal, se sustituye el producto vectorial del espacio de entrada ${x_i}^T x$ por el del espacio de características que se corresponde con el kernel:

$$h(x) = \sum_{i=1}^{n} y_i \alpha_i \, K(x_i, x) + b$$

Las funciones núcleo más utilizadas son:

- Función polinómica: está asociada a un polinomio con coeficientes a_i de propiedad conmutativa. Según el grado del polinomio, representado en i, podrá ser una función lineal (grado1), cuadrática (grado 2), cúbica (grado 3)…

  $$P(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x^1 + a_2 x^2 + \cdots + a_n x^n$$

  Particularizada para nuestro caso quedaría:

  $$K(x, x') = (x^T \cdot x' + c)^d \qquad c \in \Re\,, \qquad d \in \chi$$

- Función gaussiana: es una función definida por la siguiente expresión:

  $$f(x) = a \cdot exp\left(\frac{-(x-b)^2}{2c^2} \right)$$

  Donde a,b y c son constantes reales y a > 0. La gráfica de lafunción es una campana (campana de Gauss) donde la constante a es la altura, estando centrada en b y siendo c el ancho de la misma.
  En nuestro caso:

  $$K(x, x') = exp\left(\frac{-\|x - x'\|^2}{2\sigma^2} \right) \qquad \sigma > 0$$

- Función sigmoide: es un tipo de función que modela muchos procesos naturales y curvas de aprendizaje. Su gráfica tiene forma de S, con un inicio lento, una aceleración intermedia y final igual de suave que el principio. En general, las funciones sigmoide son funciones reales de variable real

diferenciable, con un solo punto de inflexión (es el característico su forma de S) y primera derivada no negativa.
El grupo de funciones sigmoide incluye tangentes parabólicas, arcotangentes, funciones logísticas…
En nuestro caso empleamos como ejemplo:

$$K(x, x') = \tanh(s(x^T \cdot x') + r) \qquad s, r \in \Re$$

Al hablar de kernel lineal se hace referencia al producto vectorial en el espacio de entrada que equivale a emplear la SVM de margen máximo:

$$K(x, x') = x^T \cdot x$$

Según el tipo de función kernel (polinómica, gasussiana, sigmoidal…) y de sus parámetros, se obtienen distintas fronteras de decisión. En la figura 10 se muestran dos fronteras obtenidas con una gaussiana para los mismos datos (a y b) pero con dos valores de σ (la constante c) distintos.

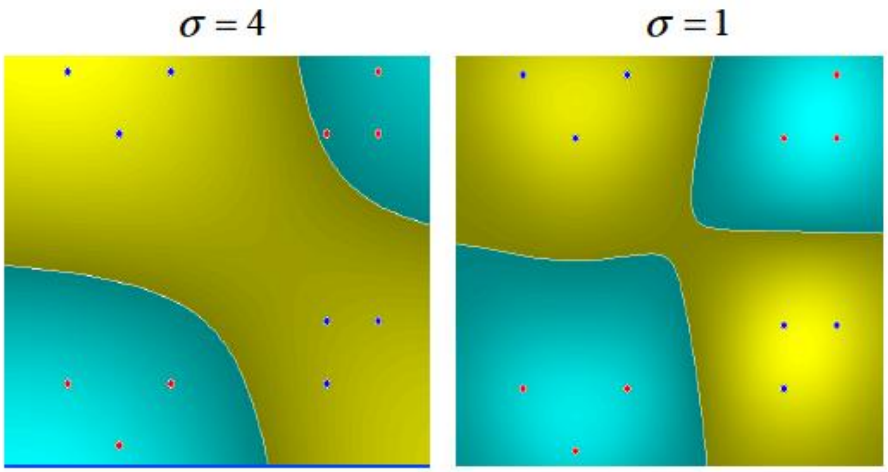


Figura 10 Fronteras de decisión obtenidas con una función núcleo gaussiana para σ=4 y σ=1

Variando el parámetro característico de la gaussiana obtenemos diferentes hiperplanos que separan los datos del ejemplo. Una de las labores al trabajar con SVM es ensayar baterías de datos con distintos valores de las funciones para obtener los que mejor se ajusten al resultado deseado. En el ejemplo vemos como con una σ=4 el hiperplano se acerca mucho a los vectores soporte de los datos rojos, mientras que con una σ=1 la separación o margen del hiperplano parece mayor.

A partir de unos datos de entrada no se puede saber cuál será el mejor kernel y los mejores valores de sus parámetros para hallar el separador óptimo. Para saber cuáles son los parámetros de la máquina de soporte vectorial más indicados se resuelve el problema para un conjunto de datos de entrenamiento y con otro conjunto de test se analiza el error. La función que se usara para la clasificación será aquella con la que se consiga la menor cantidad de puntos incorrectamente clasificados.

Generalmente, la gaussiana es la que permite obtener los separadores que mejor se adaptan a los datos y es la que se utiliza en la mayor parte de los problemas. Esto no quiere decir que no se puedan emplear otras funciones, pero la experiencia dice que el coste computacional y los resultados suelen ser favorables a la función gaussiana.

2.4.1.3 SVM LINEAL CON MARGEN BLANDO

Hay casos de datos linealmente separables en los que puede existir ruido debido a errores en la medida de los datos o por la presencia de algún outlier (dato atípico o extremo). En dichos casos no es conveniente que la SVM se ajuste totalmente a los datos.

En la figura 11 se observan dos conjuntos de datos y la frontera de decisión que se obtendría con la SVM de margen máximo. Cada conjunto está agrupado excepto por un punto que se encuentra muy próximo a los datos de la otra clase. Este se puede corresponder con un dato atípico (outlier) o que ha sido clasificado por error. Este punto no debería ser considerado para hallar la frontera de decisión ya que podría alterar los resultados deseados y nos llevaría a clasificaciones incorrectas.

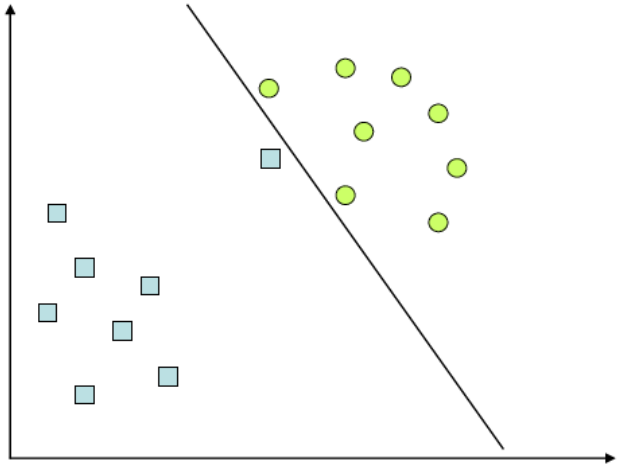


Figura 11 Clasificación con SVM de margen máximo con datos entre los que existe un outlier.

La SVM tiene que ser robusta para tener una mayor capacidad de generalización. Para ello se introducen unas variables de holgura ξ en el problema de optimización.

Partiendo de las consideraciones del problema del apartado 2.4.1.1 minimizaremos:

$$\frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n}\xi_i$$

Sujeto a las condiciones:

$$y_i(\omega^T x_i + b) \geq 1 - \xi_i \qquad 1 \leq i \leq n$$

$$\xi_i \geq 0 \qquad 1 \leq i \leq n$$

Estas condiciones ya no representan que ningún dato pueda entrar dentro del margen, si no que puede haber datos dentro del margen con una holgura ξ_i mayor que 0, gracias a lo cual podremos colocar a los datos outliers dentro de dicho margen y así evitar que distorsionen el cálculo del hiperplano.

Resolviendo el anterior problema llegaremos a la formulación dual siguiente, en la que se deberá maximizar:

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

Sujeto a:

$$\sum_{i=1}^{n} \alpha_i \, y_i = 0 \qquad 0 \leq \alpha_i \leq C \qquad 1 \leq i \leq n$$

Las variables ξ_i permiten que las restricciones no se cumplan de manera estricta: puede haber datos que cumplan $y_i h(x) < 1$. Si $y_i h(x) < 0$ significa que x está en el lado incorrecto del hiperplano.

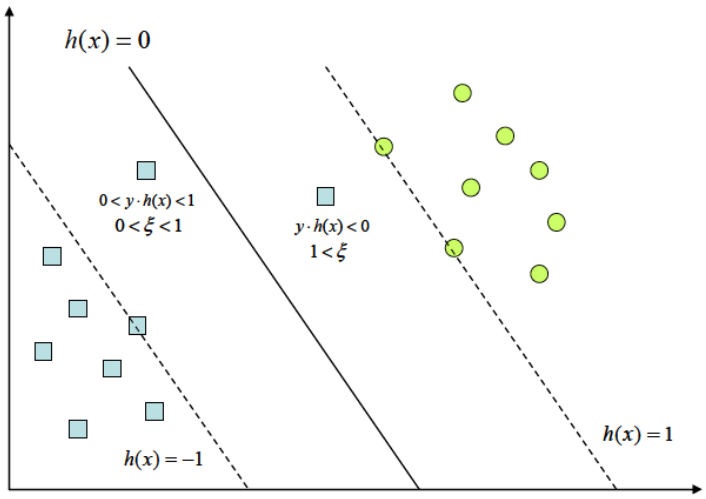


Figura 12 Clasificación de SVM con margen blando

El resultado mostrado en la figura 12 muestra un hiperplano distinto al hallado con anterioridad. Los datos extremos (en este caso dos de la clase -1) quedan dentro del margen.

La diferencia con la SVM de margen máximo es que α no puede ser mayor que C. Este parámetro permite controlar el número de errores de clasificación permitidos en la etapa de aprendizaje. Cuanto mayor es C menos ejemplos de entrenamiento serán mal clasificados. La SVM de margen máximo se corresponde con el caso en el que $C = \infty$.

Los datos x para los que $y_i h(x) = 1$ cumplen $\alpha < C$, mientras que para los que $y_i h(x) < 1$, $\alpha = C$. Todos ellos son considerados vectores soporte.

En la Figura 10 se puede observar un caso real en donde se ha ajustado un SVM con kernel lineal y varios valores de C. Los vectores soporte de este problema son:

- Para $C = \infty$: los vectores 9, 10 y 16.

- Para $C = 2$ y $C = 0{,}2$: los vectores 6, 9, 10 y 16.

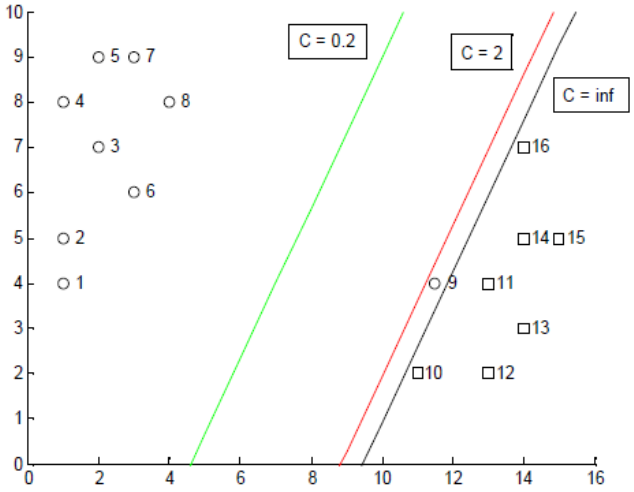


Figura 13 Fronteras de decisión para distintos valores de C.

2.4.2 MÁQUINAS DE SOPORTE VECTORIAL PARA CLASIFICACIÓN MULTICLASE

El uso habitual de las SVM es en problemas binarios. En este proyecto, cada modelo representa una categoría distinta, ya que los datos en lectura real del sensor solo coinciden con un modelo dado, por lo que tendremos tantas clases como modelos creados.

Una de las soluciones para resolver este problema multiclase es convertirlo en varios binarios. Para ello, existen 2 métodos distintos:

- Clasificación 1-v-r (del inglés one-versus-rest): en cada uno de los problemas se considera una clase positiva y las demás negativas, por lo que habrá que hallar tantos hiperplanos como clases existan.

- Clasificación 1-v-1 (del inglés one-versus-one): para cada problema se toman 2 clases de las K totales. Se compara cada clase con cada una de las restantes, lo que supone realizar K(K - 1)/2 clasificaciones.

El sistema seguido en este trabajo es el primer tipo 1-v-r ya que es el más habitual y en el que se realizan menos comparaciones, lo que se traduce en menos gasto computacional.

CAPITULO 3. APROXIMACIÓN AL RECONOCIMIENTO DE LOCALIZACIONES CON SVM

3.1 CALIBRACIÓN DEL MODELO

En el presente proyecto se busca demostrar la validez de las máquinas de soporte vectorial para clasificar superficies. Dichas superficies o ubicaciones serán conjuntos de datos no separables linealmente por lo que necesitaremos transformar dichos datos y llevarlos a un espacio de dimensión superior para poder separarlos y clasificarlos.

En el capítulo 2 se explicaron los distintos tipos de kernels disponibles. Cada una de estas funciones tiene a su vez parámetros que permiten configurarla por lo que para los diferentes casos que se den se deberá comprobar en qué medida afecta al resultado cada parámetro y optimizar su valor.

Para ello se parte del mapa en planta de la universidad. Para la primera aproximación al problema elegimos el elemento 'pasillo', siendo la tarea a realizar por el programa que modela al robot y al sensor la distintición entre datos correspondientes a un pasillo y datos que no corresponden con un pasillo.

En la figura 14 se representa el pasillo de la universidad. Marcado con recuadros rojos están las zonas denominadas 'pasillo'. Son espacios estrechos con puertas de despachos y laboratorios a los lados que están delimitadas por zonas más anchas. Estas zonas anchas marcadas con circulos azules son los 'halls' que aparte de ser más anchas dan acceso a otras ubicaciones como rellanos, ascensores, cuartos de baño...

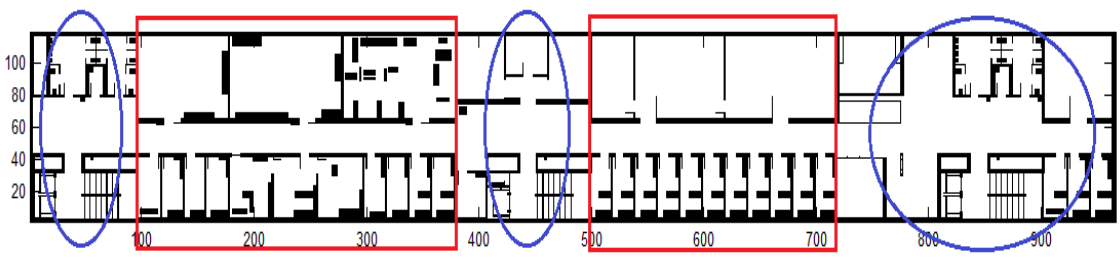


Figura 14 Mapa en planta de la universidad resaltando los pasillos

Se modelará la zona pasillo y se probarán las siguientes situaciones:

- El robot está en un pasillo, ¿clasifica los datos del sensor correctamente como pasillo?
- El robot no está en un pasillo, ¿clasifica los datos como no pasillo?

Adicionalmente se probarán los resultados con un nivel de ruido (obstáculos) de entre el 0 y el 10% y con un mapa que tenga las puertas de los despachos abiertas y otro con ellas cerradas.

El nivel de ruido representará el hecho de que las ubicaciones no siempre estarán igual que en el momento en que se tomaron los datos de entrenamiento. Puede haber personas por el medio, cajas, mobiliario nuevo… que no deben distorsionar los datos de entrada y el sistema debe ser capaz de seguir clasificando correctamente su situación con niveles bajos de ruido.

3.1.1 MODELADO

En primer lugar se debe definir qué aspectos del pasillo se modelarán. En el caso que aplica se decide tener en cuenta las dos paredes y el espacio correspondiente a las puertas y parte del despacho. Si no se modelan las puertas y una parte de los despachos, en caso de que el robot recorra el pasillo y tome medidas cerca de una puerta abierta daría un resultado anómalo porque más del 20% de las medidas no sabría clasificarlas.

Dada la forma del plano, se decide no modelar el interior de los laboratorios porque están vacíos. En un caso real habría que evaluar el hecho de que pueda haber estanterías, mesas u obstáculos que pudiera reconocer el sensor.

En la figura15 se muestra en rojo sobre el plano la parte modelada correspondiente al pasillo.

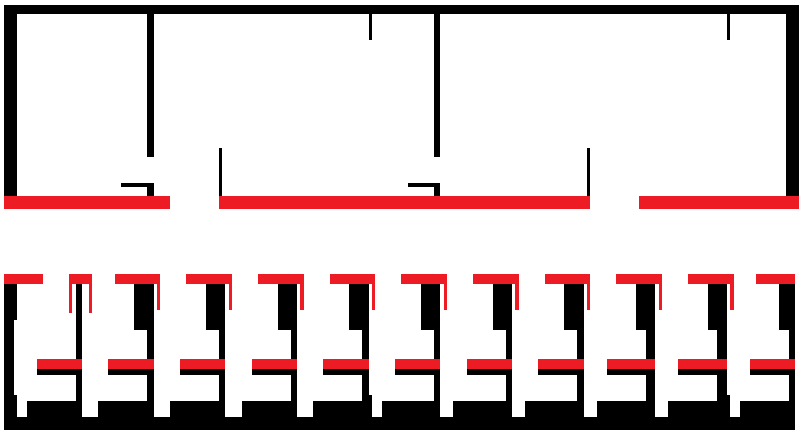


Figura 15 Zonas a modelar dentro del espacio ‘pasillo’

Captamos suficientes vectores de entrenamiento y se etiquetan (1) los datos correspondientes a un pasillo y se crea un número adecuado de resultados no-pasillo (0) para contrastar el modelo.

De los datos de entrenamiento se emplearán un 75% para caracterizar el modelo y el restante 25% para probarlo y si es necesario mejorar la profundidad del modelo para aumentar su fiabilidad.

El proceso de aprendizaje mediante la herramienta creada en Matlab se explicará en detalle en el capítulo 4 y el manejo de la herramienta en el anexo correspondiente. En este apartado de calibración se adquieren los datos con la herramienta principal y se trabaja sobre ellos de forma manual en Matlab para poder adaptarlos de manera individual. A continuación se crearán distintos modelos empleado los diferentes kernels existentes en la toolbox Lisvm.

Respecto al etiquetado, en la teoría de las SVM se emplea el valor 1 para designar a la etiqueta positiva y el valor -1 para designar la etiqueta negativa pero la librería empleada en Matlab utiliza las etiquetas 1 y 0 respectivamente. En la figura 16 quedan representado los datos según el etiquetado, los círculos representan a la etiqueta 1 (pasillo) y las cruces a la etiqueta 0 (no-pasillo).

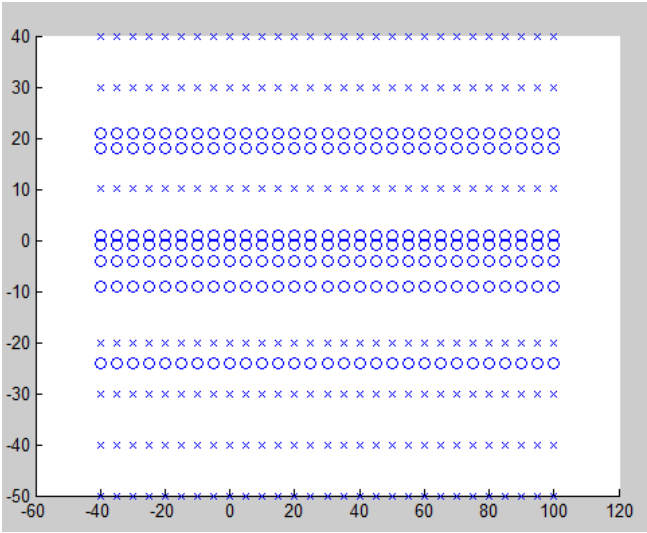


Figura 16 Datos correspondientes a un pasillo

En este ejemplo partiocular se observa como las medidas son constantes en el eje Y, por lo que el modelo es estrapolable a cualquier punto del pasillo. Las etiquetas negativas se han elegido de manera que fuercen a los kernel a ajustarse alrededor de los datos etiquetados positivamente como pasillo, por lo que se han intercalado entre éstos.

Una vez que el etiquetado está definido, se deberá elegir la función kernel que separe los datos con mayor precisión y posteriormente ajustar dicha función.

3.1.2 ESCALADO DE DATOS

Siempre es recomendable escalar los datos antes de operar con ordenadores. Para un ordenador, la operación $10^5 \times 10^5$ es mucho más costosa (en términos computacionales) que $10^2 \times 10^2$. En el presente proyecto la librería Libsvm daba resultados anómalos cuando los datos estaban comprendidos en el intervalo [0,1] o en el [-1,-1] (ejemplo típicos de intervalos reducidos) por lo que se decide mantener los valores de entrada absolutos.

Este caso presenta valores de un orden máximo de centenas y en los resultados del proyecto los tiempos de cálculos han sido mínimos, por debajo de décimas de segundo.

Si el proyecto estuviese orientado al reconocimiento de frecuencias de sonido o de luz (datos con un orden por encima del millar) o similares el escalado sería obligatorio ya que no se podría garantizar que empleando un ordenador de bajas características los tiempos de cálculo se mantuvieran por debajo de las décimas de segundo.

3.1.3 ELECCIÓN DEL TIPO DE SVM

Dentro del método de las máquinas de soporte vectorial existen dos clases, las de clasificación o las de regresión. Originalmente las SVM se crearon para resolver problemas de clasificación, pero se ampliaron posteriormente para problemas de regresión.

Las de tipo clasificación se emplean para obtener resultados de tipo cualitativo, por ejemplo, determinar la clase de un dato de entrada, mientras que las de tipo regresión son más útiles en problemas cuantitativos, cuando se trata de obtener una salida numérica al dato de entrada.

En este proyecto solo se emplearán máquinas de soporte vectorial de tipo clasificación ya que el objetivo buscado es obtener la clase a la que pertenecen los datos de entrada.

La librería Livsum, con la que se trabaja, ofrece también la opción de SVM de regresión (con los tipos epsilon-SVR y nu-SVR).

Dentro de las máquinas de soporte vectorial clasificadoras tenemos dos tipos, como se explicó en el capítulo 2:

- **SVM Tipo 1 o C-SVM**

Este tipo de máquina de soporte vectorial permite que haya datos dentro del margen, con una holgura ξ_i mayor que 0, gracias a lo cual los datos outlier no distorsionan el cálculo del hiperplano.

El parámetro C permite controlar el número de errores de clasificación permitidos en la etapa de aprendizaje. Cuanto mayor es C menos ejemplos de entrenamiento serán mal clasificados. La SVM de margen máximo se corresponde con el caso en el que $C = \infty$.

Por lo tanto el problema de clasificación es el resultante de minimizar:

$$\frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n}\xi_i$$

Con las condiciones:

$$y_i(\omega^T x_i + b) \geq 1 - \xi_i \qquad 1 \leq i \leq n$$

$$\xi_i \geq 0 \qquad 1 \leq i \leq n$$

Resolviendo el anterior problema llegaremos a la formulación dual siguiente, en la que se deberá maximizar:

$$\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

Sujeto a las siguientes condiciones donde la aparece el parámetro C

$$\sum_{i=1}^{n}\alpha_i\, y_i = 0 \qquad 0 \leq \alpha_i \leq C \qquad 1 \leq i \leq n$$

- **SVM Tipo 2 o NU-SVM**

En el segundo tipo de SVM se mantiene la holgura ξ_i pero en vez del parámetro C se maneja el parámetro nu (ν) que también permite cierto control sobre el número de datos de entrenamiento y vectores soporte. Dicho parámetro ν ϵ (0, 1]

La formulación consiste en minimizar:

$$\frac{1}{2}\omega^T\omega - \upsilon\rho + \frac{1}{n}\sum_{i=1}^{n}\xi_i$$

Sujeto a las condiciones:

$$y_i(\omega^T x_i + b) \geq \rho - \xi_i \qquad 1 \leq i \leq n$$

$$\xi_i \geq 0 \qquad 1 \leq i \leq n$$

$$\rho \geq 0$$

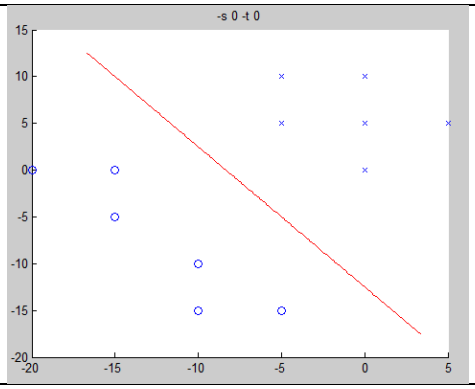
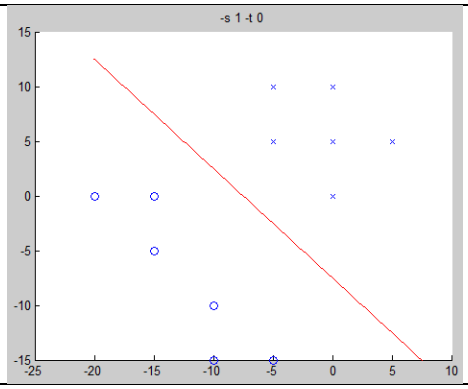
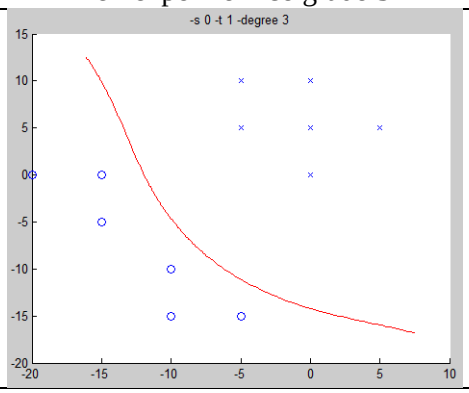
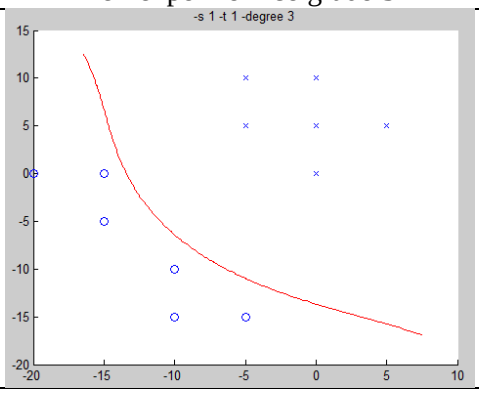
Se ha definido el uso de la máquina de soporte vectorial tipo clasificación ya que los resultados buscados en el proyecto son cualitativos.

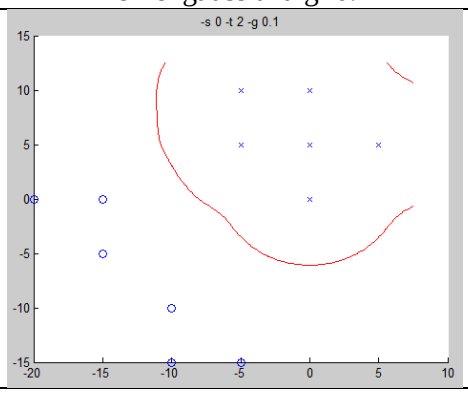
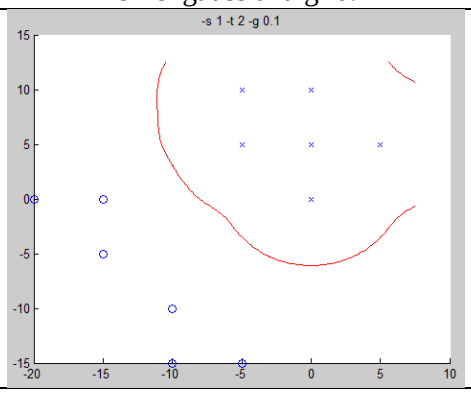
Conocidos los dos tipos de SVM tipo clasificación, se procederá a ensayar para distintos conjuntos de datos los dos tipos de algoritmos para ver cual se ajusta mejor.

Los datos elegidos para el ensayo serán similares a los que nos encontraremos en la realización del proyecto. Tendrán una distribución uniforme sobre el espacio de estudio y existirán datos outliers aunque a la hora de crear modelos en el proyecto definitivo se eliminarán.

En primer lugar se ensayarán datos linealmente separables y se probarán tres tipos de kernels. El empleo de los kernels es para darle profundidad al ensayo ya que la elección del tipo de kernel y su parametrización se elegirá después de tener decidido el tipo de SVM.

Tabla 1 Comparativa resultados C-SVM frente a NU-SVM con datos linealmente separables

C-SVM	**NU-SVM**
Kernel lineal 	Kernel lineal 
Kernel polinómico grado 3 	Kernel polinómico grado 3 

Kernel gaussiana g=0.1	Kernel gaussiana g=0.1
	

Como se comprueba en la tabla 1, se probaron a modo de ejemplo un kernel tipo lineal, uno polinómico de grado 3 y uno gaussiano.

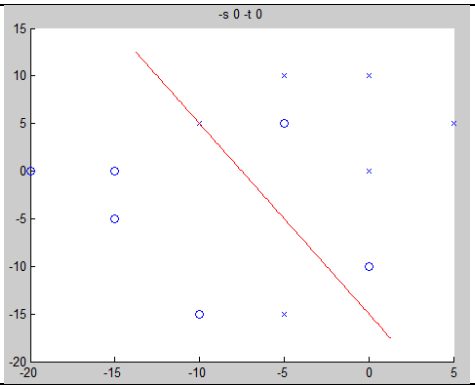
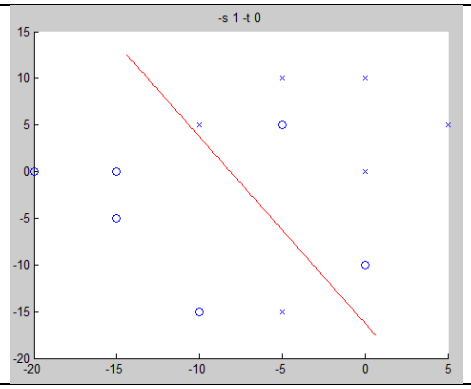
Al ser datos linealmente separables no sería necesario el empleo de kernels ya que en el espacio de dimensión dos se pueden separar dichos datos sin necesidad de transformaciones previas.

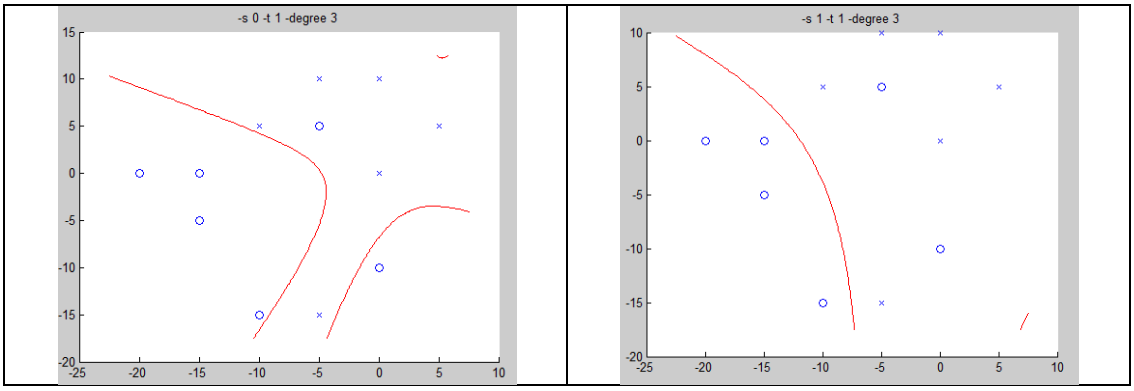
El resultado es satisfactorio tanto con la C-SVM como con la NU-SVM y el kernel lineal es el que parece que mejor se ajusta para separar las dos clases. Otra ventaja es el menor coste computacional ya que al ser una recta solo se configuran dos parámetros.

A continuación se ensayan datos que no son linealmente separables y los mismos 3 tipos de kernel.

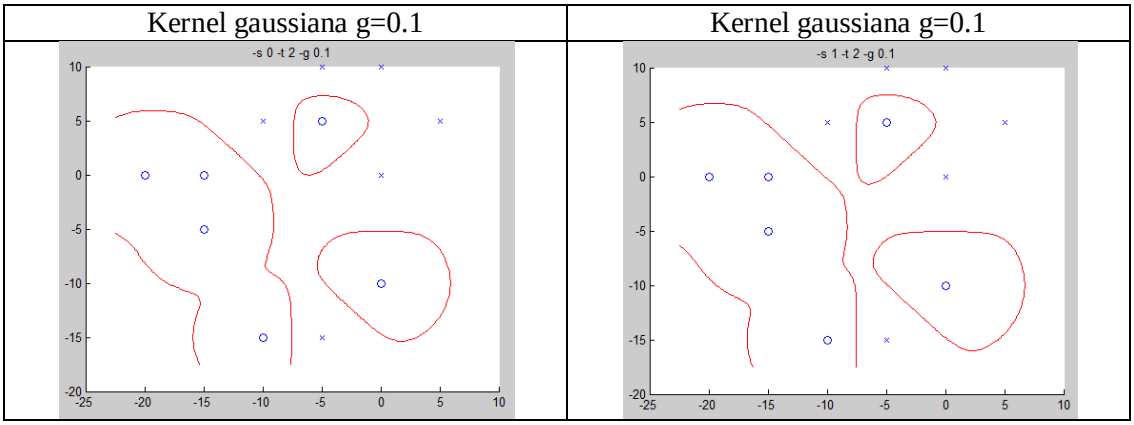
Las simulaciones se representan en la tabla 2:

Tabla 2 Comparativa resultados C-SVM frente a NU-SVM con datos linealmente separables

C-SVM	NU-SVM
Kernel lineal	Kernel lineal
	
Kernel polinómico grado 3	Kernel polinómico grado 3



Kernel gaussiana g=0.1	Kernel gaussiana g=0.1



En el caso de datos no separables linealmente ya es preciso el empleo de kernels que transformen los datos a un espacio de dimensión mayor donde sean separables por un hiperplano.

El empleo de tres kernels distintos es un anticipo de los resultados que obtendremos en el siguiente apartado.

Se comprueba que los resultados son muy similares, tanto con la C-SVM como en la NU-SVM, y que no hay un algoritmo que destaque, pero lo que si se va comprobando es la ventaja de emplear un kernel con función gaussiana respecto a los otros en el caso de datos no linealmente separables.

Como consecuencia de las pruebas se opta emplear una **C-SVM** ya que los resultados son muy similares a la NU, pero tenemos más información de la primera.

3.1.4 ELECCIÓN DEL TIPO DE KERNEL

La librería Libsvm nos ofrece la posibilidad de emplear cuatro familias de funciones kernel, cada una personalizable con parámetros de precisión y de distribución espacial. Estas cuatro funciones son las mismas que se desarrollaron en el capítulo 2.

- Función polinómica: está asociada a un polinomio con coeficientes a_i de propiedad conmutativa. Según el grado del polinomio, representado en i, podrá ser una función lineal (grado1), cuadrática (grado 2), cúbica (grado 3)…

$$P(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x^1 + a_2 x^2 + \cdots + a_n x^n$$

Particularizada para nuestro caso quedaría:

$$K(x, x') = (x^T \cdot x' + c)^d \qquad c \in \Re\,, \qquad d \in \chi$$

- Función gaussiana: es una función definida por la siguiente expresión:

$$f(x) = a \cdot exp\left(\frac{-(x-b)^2}{2c^2}\right)$$

Donde a,b y c son constantes reales y a > 0. La gráfica de la función es una campana (campana de Gauss) donde la constante a es la altura, estando centrada en b y siendo c el ancho de la misma.
En nuestro caso:

$$K(x, x') = exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right) \qquad \sigma > 0$$

- Función sigmoide: es un tipo de función que modela muchos procesos naturales y curvas de aprendizaje. Su gráfica tiene forma de S, con un inicio lento, una aceleración intermedia y final igual de suave que el principio. En general, las funciones sigmoide son funciones reales de variable real diferenciable, con un solo punto de inflexión (es el característico su forma de S) y primera derivada no negativa.
El grupo de funciones sigmoide incluye tangentes parabólicas, arcotangentes, funciones logísticas…
En nuestro caso empleamos como ejemplo:

$$K(x, x') = \tanh(s(x^T \cdot x') + r) \qquad s, r \in \Re$$

Se parte de los datos etiquetados en el apartado 3.1.1. Estos datos previamente habían sido guardados en un fichero .mat de Matlab. Se recuperan y se corren a través de la función svmtoy (2).

La función svmtoy utiliza las funciones de SVM de Libsvm, pero genera la representación gráfica de estás, siendo más vistoso el resultado, ya que con Libsvm solo tenemos una salida en el *Command Window* mostrando los datos del modelo generado cuando entrenamos o el etiqueta con su fiabilidad en la clasificación.

Los resultados obtenidos para las cuatro funciones kernel con sus ventajas y desventajas se muestran a continuación:

3.1.4.1 KERNEL LINEAL

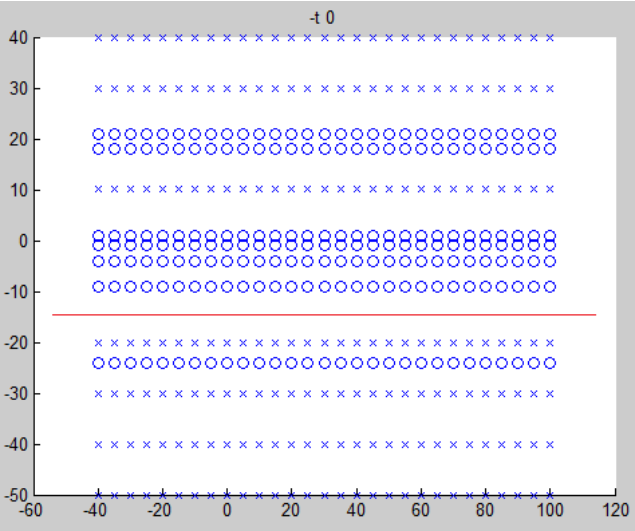


Figura 17 Pasillo modelado con kernel lineal

Una función lineal no es capaz de resolver el hiperplano que separa un conjunto de datos no separables linealmente. La línea roja representa el hiperplano obtenido y se observa como no cumple el objetivo propuesto.

El resultado de un kernel lineal es idéntico al obtenido por una función polinómica de grado 1.

Ventajas

- Velocidad de cálculo

Desventajas

- Resultado muy limitado. En un caso de datos no separables linealmente no se genera un hiperplano correcto.

3.1.4.2 KERNEL POLINÓMICO DE GRADO 2

El hiperplano obtenido por un kernel polinómico de grado 2 tampoco es un resultado positivo, pero mejora sensiblemente al obtenido por un kernel lineal o polinómico de grado 1.

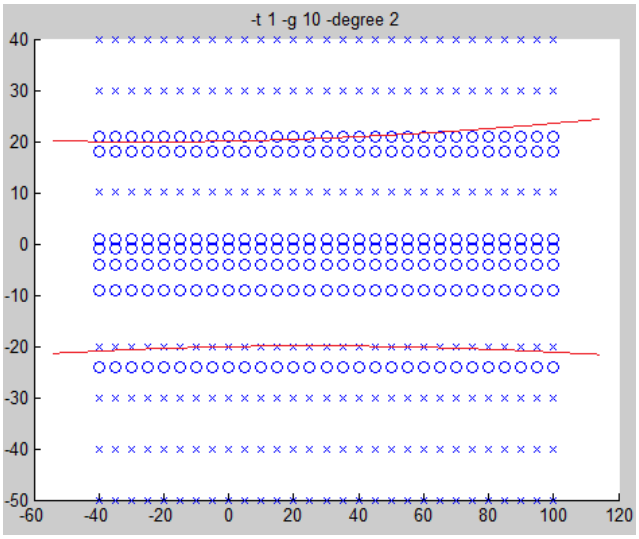


Figura 18 Pasillo modelado con kernel polinómico de grado 2

Ventajas

- No se encuentran

Desventajas

- Resultado muy limitado. En un caso de datos no separables linealmente no se genera un hiperplano correcto.
- Cuanto mayor es el grado del polinomio, mayor es el tiempo de cálculo, por encima de grado 3 y teniendo más de 400 datos de entrada para clasificar los tiempos de espera superan el minuto.

3.1.4.3 KERNEL GAUSSIANO

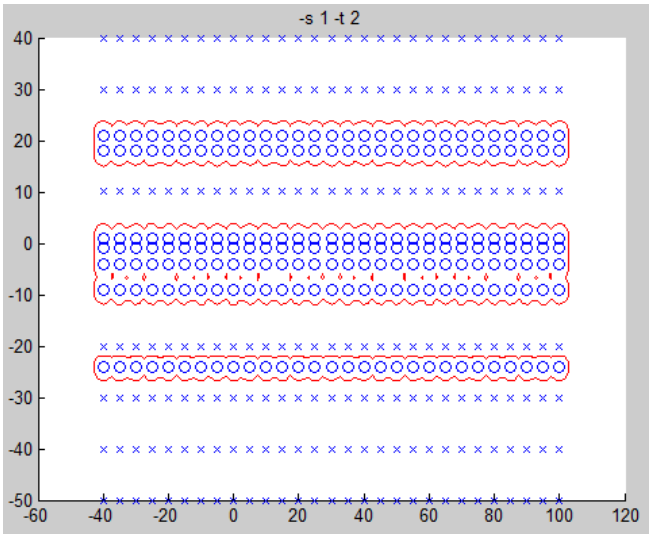


Figura 19 Pasillo modelado con kernel gaussiano

La función gaussiana si es capaz de transformar los datos de forma que el hiperplano se ajusta correctamente a la clase positiva separándola de la negativa.

Ventajas

- Resultado satisfactorio
- Tiempo de cálculo inapreciable (menor a décimas de segundo).

Desventajas

- No se encuentran

3.1.4.4 KERNEL SIGMOIDAL

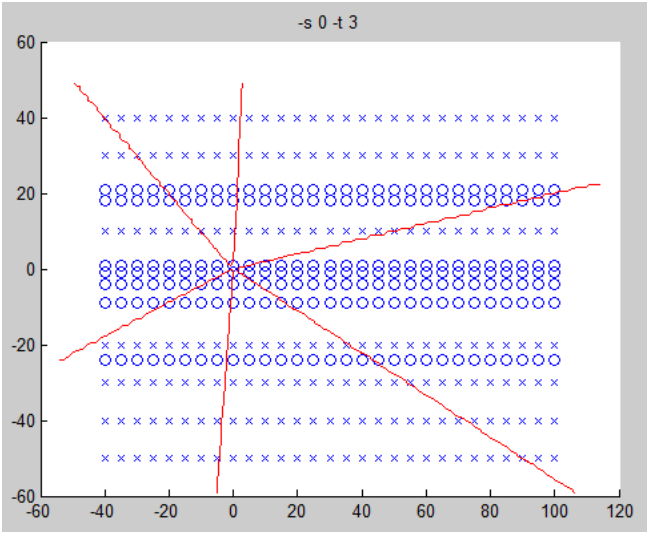


Figura 20 Pasillo modelado con kernel sigmoidal

Al contrario que con la función gaussiana, la función sigmoide no es válida para su empleo con datos no separables linealmente.

Ventajas

- Velocidad de cálculo

Desventajas

- Resultado insatisfactorio.

Con los 4 tipos probados se decide emplear una gaussiana como función de kernel para la separación de los datos, ya que es la que ofrece los mejores resultados.

3.1.5 PARAMETRIZACIÓN DE LA FUNCIÓN KERNEL

Analizando los resultados gráficos de las pruebas del modelo de un pasillo con los cuatro tipos de funciones kernel disponibles en la librería Libsvm se decide emplear la función gaussiana ya que es la que ofrece el resultado más satisfactorio.

Las funciones polinómica de grados altos también arrojaron resultados correctos, pero su tiempo de cálculo es muy superior al de la gaussiana, siendo éste el motivo por el cual se descartaron.

La variable fundamental de esta función particularizada para nuestro caso es la σ que se corresponde con la c en su formulación general.

$$K(x, x') = exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right) \qquad \sigma > 0$$

La σ en una campana de Gauss es la anchura de la campana. La función anterior es de altura la unidad y estaría centrada en x' que es un dato de entrada, por lo que el único parámetro modificable de antemano es la σ.

Para conocer en que medida afectar este valor a nuestro resultado se emplearán tres valores de sigma con una separación de una década entre ellos (10, 1 y 0.1) y se valorará cual es el más adecuado.

En primer lugar se ensaya con una $\sigma = 10$. En este caso el hiperplano solo rodea a los datos etiquetados con 1 (la clase deseada) y perderíamos fiabilidad para clasificar datos que estén próximos a ellos.

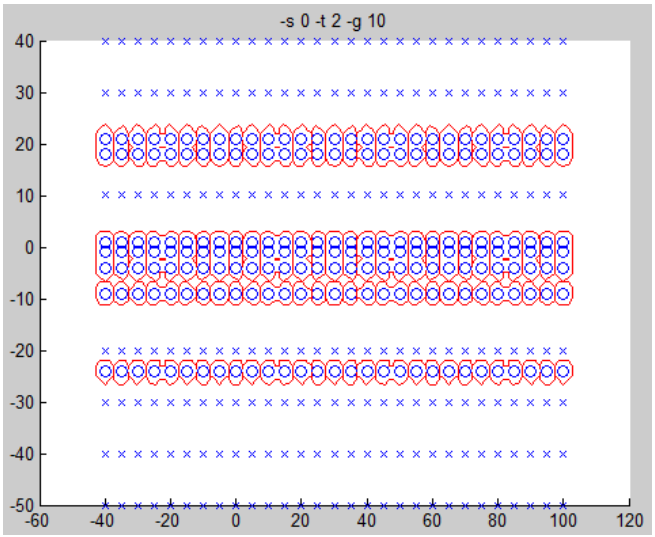


Figura 21 Pasillo modelado con una gaussiana con $\boldsymbol{\sigma = 10}$

A mayores valores de σ la función se ajusta más a los valores y esto no es del todo

deaseable ya que perdemos la posibilidad de generalización que tendríamos en caso de que en vez de ajustarse los rodeara a mayor distancia (siempre dentro de los límites que fuerzan las etiquetas negativas).

En segundo lugar se ensaya con una $\sigma = 1$ y se obtiene un resultado mejor que con $\sigma = 10$ pero sigue sin ser del todo satisfactorio ya que sigue ajustándose en exceso.

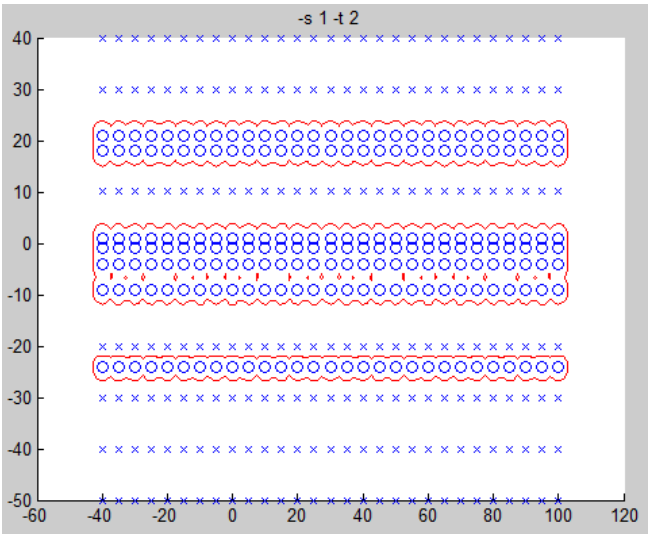


Figura 22 Pasillo modelado con una gaussiana con $\boldsymbol{\sigma = 1}$

Por último probamos una σ menor, del orden de 0.1, que nos ofrece una separación de datos más intesante, de forma que aporta un margen suficiente para etiquetar datos próximos a los de entrenamiento.

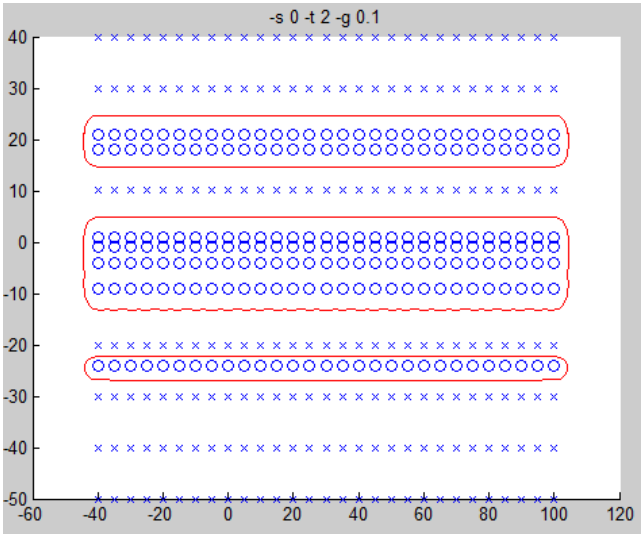


Figura 23 Pasillo modelado con una gaussiana con $\boldsymbol{\sigma = 0.1}$

En base a los resultados obtenido se decide emplear una $\sigma = 0.1$ para el modelado de los espacios del proyecto. A menor σ el hiperplano resulta más holgado y obtendremos clasificaciones positivas en datos próximos.

En otro tipo de problemas no sería deseable porque esto puede dar lugar a errores y confusiones en clasificaciones cuando dos modelos son muy similares, por ejemplo si queremos diferenciar dos formas de cara humana mediante SVM. En un problema así lo importante sería modelar correctamente los detalles y deberíamos emplear σ mayores a costa de utilizar un mayor número de vectores de entrenamiento.

Sin embargo en el presente proyecto, los espacios modelados difieren mucho el uno del otro por lo que no es necesario tal nivel de precisión.

En la librería empleada tenemos otros parámetros para ajustar el modelo, estas son gamma y en el caso de la máquina de soporte vectorial empleada, C, que permite aumentar o disminuir el margen de separación. Si se hubiera utilizado el tipo 2 de SVM (NU-SVM) en vez de C utilizaríamos el parámetro nu.

Por defecto al realizar el entrenamiento los valores de C y gamma son 1.

Se realizaron pruebas variando estos valores y en los casos probados, modelando el pasillo, los resultados no difirieron de los obtenidos con los valores por defecto.

3.2 ENSAYO DEL MODELO PASILLO

Una vez creado el modelo del pasillo debemos contrastarlo para comprobar su validez. Esto se hará con una batería de datos de entrenamiento, aproximadamente el 25%, que en este caso se ha completado con medidas realizadas en zonas de no pasillo para comprobar si el algoritmo clasifica correctamente datos que no están en un pasillo.

Para ello situamos el robot en diversas localizaciones, 6 en zonas de pasillo y 6 en zonas de no-pasillo. Sobre las figuras 24 y 25 están representados los lugares de adquisición de los datos.

Figura 24 Posiciones del sensor zonas pasillo / no pasillo (1/2)

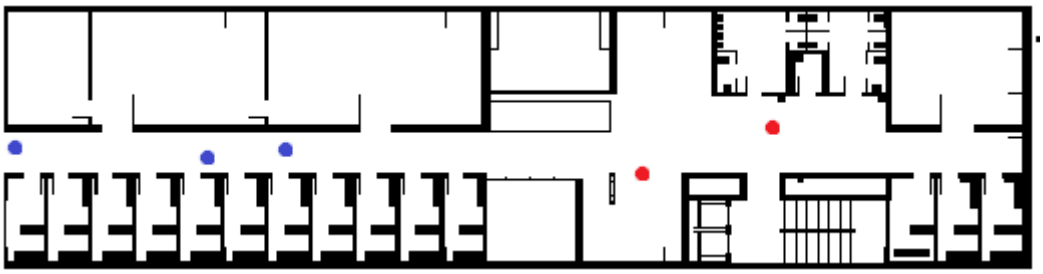


Figura 25 Posiciones del sensor zonas pasillo / no pasillo (1/2)

En primer lugar comprobamos el resultado de clasificar las 6 medidas (2 en cada hall) de las zonas no pasillo

Tabla 3 Clasificación medidas no-pasillo

Medida	Fiabilidad
1	52.071% (88/169)
2	48.5207% (82/169)
3	52.5% (84/160)
4	56.8047% (96/169)
5	32.0755% (51/159)
6	39.881% (67/168)

Ninguna de las lecturas del sensor es clasificada como pasillo con más del 60% de fiabilidad. Al ser un ejemplo sencillo, comprobamos que estando en un hall aproximadamente la mitad de las medidas corresponderán a una de las paredes que es la misma del pasillo, por ello la fiabilidad ronda el 50%.

Toda medida que contenga datos sobre una pared, más o menos larga, tendrá una clasificación bastante alta cuando la comparamos con un modelo de pasillo, al tratarse este de dos paredes paralelas. Esto será inevitable por lo que al contrastar este tipo de medidas se le exigirá una fiabilidad elevada, por encima del 90%, para clasificar con seguridad los datos.

A continuación colocamos el sensor en zonas de pasillo y empleamos un mapa con las puertas abiertas, una situación desfavorable ya que parte de las medidas podrán fugarse a través de las puertas y 'rebotar' en zonas de despacho, laboratorios… Siendo este hecho límite cuando el sensor del robot se encuentre enfrente de una puerta abierta.

Tabla 4 Clasificación medidas tipo pasillo con mapa puertas abiertas

Medida	Fiabilidad
1	100% (169/169)
2	100% (169/169)
3	100% (169/169)
4	95.858% (162/169)
5	97.619% (164/168)

6	97.6331% (165/169)

El algoritmo clasifica las 6 medidas como ‘pasillo’ con fiabilidades por encima del 97% lo que valida el modelo. En las medidas 4, 5 y 6 la fiabilidad baja del 100% ya que alguno de los puntos se sale del pasillo hacia los despachos.

Anteriormente se explicó que al comparar las medidas con datos tipo pasillo se exigirían fiabilidades por encima del 90% y se comprueba que en este caso las 6 lecturas serían correctamente clasificadas como pasillo.

A continuación se repiten las medidas con todas las puertas cerradas.

Tabla 5 Clasificación medidas pasillo con mapa puertas cerradas

Medida	**Fiabilidad**
1	100% (169/169)
2	100% (169/169)
3	100% (169/169)
4	100% (169/169)
5	100% (169/169)
6	100% (169/169)

En este caso ideal todas las medidas son clasificadas en un 100% como ‘pasillo’ ya que con las puertas cerradas el pasillo queda definido por dos paredes paralelas lo que le otorga una similitud del 100% respecto al modelo creado.

Por último emplearemos un mapa con cierto nivel de ruido representado por obstáculos (figura 26). Los obstáculos son de cierto tamaño, 70x30 cm aproximadamente para comprobar la incidencia de estos en la clasificación de las medidas.

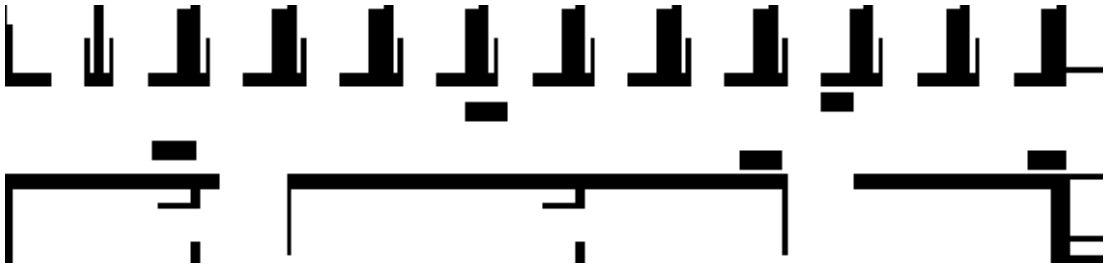


Figura 26 Mapa del pasillo con obstáculos

Se repiten las 6 medidas con el nuevo mapa y se obtienen los siguientes resultados:

Tabla 6 Clasificación medidas pasillo con mapa obstáculos y puertas abiertas

Medida	Fiabilidad
1	96.5714% (169/175)
2	91.716% (155/169)
3	78.0347% (135/173)
4	90.2857% (158/175)
5	90.7514% (157/173)
6	95.3757% (165/173)

Se observa una disminución de la fiabilidad en la clasificación que es proporcional al número de medidas que en cada lectura 'chocan' contra una obstáculo.

Conforme el sensor esté más próximo a un obstáculo, éste cegará más medidas que en el caso de estar más alejado como se muestra en la figura 24.

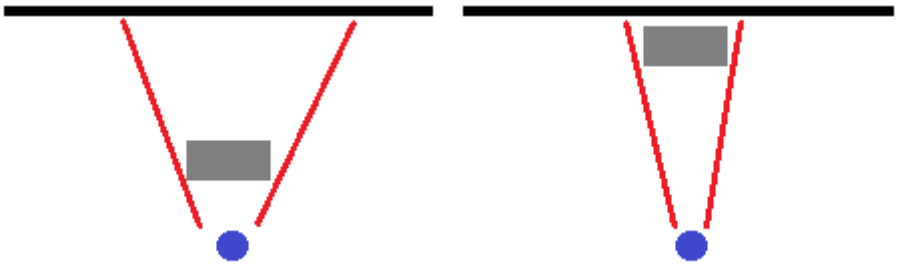


Figura 27 Posiciones de un obstáculo respecto al sensor

La cuestión de los obstáculos resulta complicada de resolver para un análisis estático del sensor.

Si colocamos el sensor en un punto del espacio que resulta que está muy próximo a un obstáculo la medida estará distorsionada y el resultado no coincidirá con ningún modelo. Esto es un escollo insalvable que dará algún dato anómalo en el próximo capítulo.

Si pensamos que el sensor irá montado en un robot en movimiento, dicho robot se acercará o alejará de los obstáculos por lo que partiendo de una situación en la que pueda ubicarse, todo movimiento que lo acerque a un obstáculo lo captará gracias a la pérdida de fiabilidad en la clasificación y podrá subsanarlo corrigiendo el rumbo o retrocediendo.

Con los ensayos realizados se valida el método de las SVM para el reconocimiento de ubicaciones. Las fiabilidades en las clasificaciones de los pasillos con lecturas de

pasillos son muy elevadas, mientras que los contrastes al tomar las lecturas en zonas no pasillo son correctos.

En el siguiente capítulo se extenderá el método a las ubicaciones más representativas de la universidad mediante la programación de una interfaz que nos permita recorrerla con un robot simulado y clasificarlas contra los modelos que se establezcan.

CAPITULO 4. APLICACIÓN DEL RECONOCIMIENTO DE LOCALIZACIONES CON SVM

4.1 PROGRAMACIÓN DE LA INTERFAZ GRÁFICA

Una vez comprobada la validez de las SVM para la clasificación de espacios partiendo de los datos adquiridos por un sensor procedemos a realizar un programa en Matlab con el que simular las distintas situaciones que se puedan dar en nuestro estudio.

El programa tendrá como punto de partida el mapa en planta de un edificio de la universidad y diversas opciones para clasificar los datos.

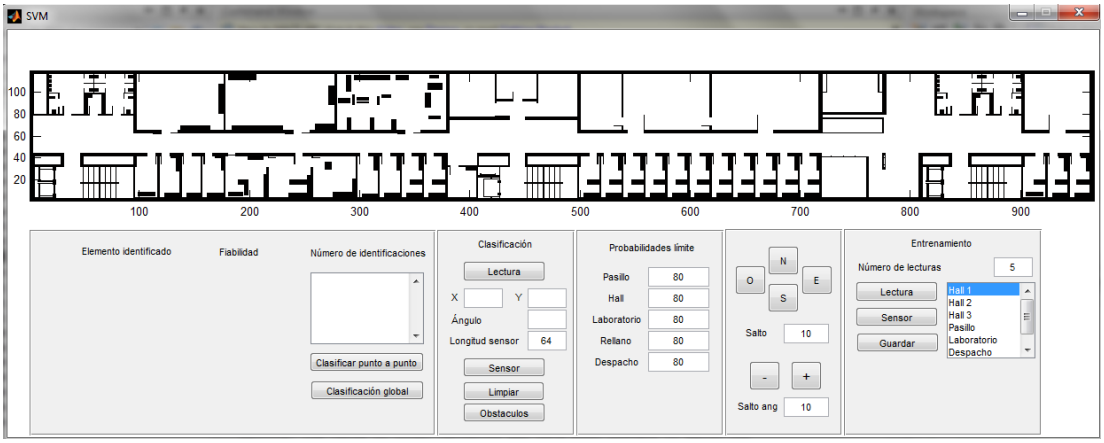


Figura 28 Programa en Matlab

En la interfaz gráfica realizada con la herramienta *GUIDE* de Matlab aparece el mapa en la parte superior. Sobre el mapa podremos pinchar y situar tanto la posición del robot como la dirección del sensor para realizar las medidas.

En la parte inferior de la pantalla tendremos las opciones de configuración formadas por 5 bloques.

El bloque de la derecha es el que se emplea en el entrenamiento del algoritmo. Consta de un *edit text* en el que se definen en número de medidas que queremos

realizar, tres botones (lectura, sensor y guardar) y un menú desplegable tipo *listbox* en el que se selecciona el tipo de localización sobre la que se está realizando el entrenamiento.

A continuación tenemos un bloque tipo *control pad* que nos permite, una vez situado el robot sobre el mapa, moverlo en las direcciones norte, sur, este y oeste y girar la dirección del sensor. Para ello consta de 4 botones de dirección, 2 botones de cambio del ángulo (+ y -) y dos *edit text* en los que definimos el salto mínimo tanto en distancia como en ángulo que se aplicará en cada movimiento.

El siguiente bloque es en el que se permite definir las fiabilidades límite para los distintos modelos. Es un dato importante, que para los casos generales viene dado al 80%, pero que en cada caso particular se podrá ajustar en función de las necesidades. Cuando operemos con el programa, si tenemos las fiabilidades al 100% cualquier medida que se salga del modelo deseado ya nos dará un resultado por debajo de ese 100% y no obtendremos una clasificación válida, por eso, se debe dejar un margen de error ya que siempre hay medidas que no encajan exactamente en los modelos.

El cuarto bloque por la izquierda, o segundo por la derecha, es el que tiene las opciones para ejecutar el algoritmo de clasificación. Consta del botón lectura, que habilita la opción de posicionar el robot sobre el mapa, del botón sensor, que ejecuta una función que simula al sensor del robot, del botón limpieza, que elimina los dibujos auxiliares sobre la imagen del mapa y vacía las variables empleadas en el programa. También tenemos un botón de nombre obstáculos, que permitirá cargar un mapa con cierto nivel de ruido para probar la validez de clasificación con datos mezclados con ruido.

En este bloque, cada vez que situamos al sensor se rellena los cuadros *edit text* con la situación espacial del robot (en x e y) y el ángulo de medida. Además hay otro campo para definir la distancia máxima medida por el sensor, ya cada tipo de sensor tiene un alcance máximo dependiendo de la tecnología empleada.

El último bloque saca a pantalla los resultados de la clasificación que se obtienen en función de si ejecutamos la ‘clasificación punto a punto’ o la ‘clasificación global’. Estos dos botones llaman a la función que resuelve el mismo problema de clasificación pero mientras que punto a punto podemos ver el resultado después de comparar cada uno de los puntos característicos hallados por el sensor, en la global devuelve el resultado general de la clasificación de la localización.

El resultado se muestra en la columna de ‘elemento identificado’ y la fiabilidad de dicho resultado en la columna situada a su derecha, siendo ésta siempre superior a las fiabilidades límites que se habían definido previamente.

El código completo se encuentra en el CD adjunto al proyecto.

El procedimiento para simular la actuación del sensor consiste en emplazarlo en un punto del mapa y una vez ubicado hay que indicar la dirección en la cual toma la medida. La dirección queda representada por una línea que une los dos puntos dibujados sobre el mapa. El sensor tendrá un alcance definido por el usuario. En nuestro caso empleamos uno de 64 celdas. Al ser cada celda igual a 0,125 metros, el

sensor empleado tendría un alcance de 8 metros (típicamente uno de ultrasonidos). El total de puntos por medida será de 90, ya que toma medidas cada 2º teniendo una amplitud total de 180º.

Un ejemplo de una toma de medidas se ve en la siguiente figura

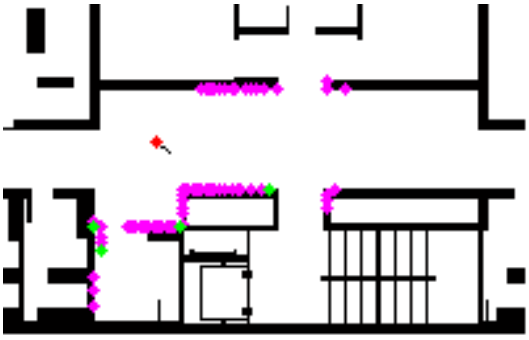


Figura 29 Ejemplo de medidas tomadas por el sensor

Se ve que hay dos tipos de puntos dibujados sobre el mapa, violetas y verdes.

El programa recibe los datos del sensor y los clasifica empleando un algoritmo que reconozca los puntos característicos más cercanos al sensor. El algoritmo va marcando con puntos verdes los que representen un cambio mayor que un margen interno, tanto en coordenadas x como y, de forma que distinga esquinas, salientes, huecos…

Dichos puntos verdes se emplean como 'orígenes' sobre los que se escalan el resto de medidas del sensor. Esto es necesario para poder comparar cada muestra de medidas con los modelos creados de los distintos espacios ya que si no sería imposible clasificar los datos ya que el sensor se puede encontrar en cualquier punto del espacio.

En otras aplicaciones de las SVM como puede ser el reconocimiento de escritura no existe este problema ya que el reconocimiento se hace sobre una hoja de papel (utilizando un sensor que sea una cámara fotográfica o un escáner, por ejemplo) y se puede emplear un origen de datos común, como puede ser la esquina superior derecha del papel, que será invariable para cualquier clasificación.

Teniendo los puntos singulares de la muestra de medidas localizados se reescalan el resto de datos tomando dichos puntos como origen y se clasifican contra los modelos que se han creado de cada zona.

La clasificación, como se comentó en el capítulo 2, será 1-v-r (del inglés one-versus-rest), esto es, en cada modelo clase positiva y las demás negativas.

Empleamos este método porque la toolbox de Matlab que utilizamos resuelve mejor los problemas binarios a pesar de que es capaz de trabajar con más de dos clases.

Una vez clasificados todos los puntos se comparan entre sí según el resultado de la clasificación que se haya obtenido para decidir en qué localización estamos.

4.2 PROGRAMACIÓN DEL CÓDIGO

El código interno del programa está basado en funciones que se relacionan con las acciones sobre los botones de la interfaz gráfica.

La programación se apoya en la librería Libsvm para los problemas de clasificación y la función sensor proporcionada por la universidad Carlos III, el resto del código fue desarrollado en exclusiva para el presente proyecto.

En el CD adjunto se encuentra todo el código y sin compilar, en formato Matlab.

4.3 DESARROLLO DE LOS MODELOS

Se crearon modelos de siete espacios singulares del plano, representados con ejemplos en las figuras 30 y 31.

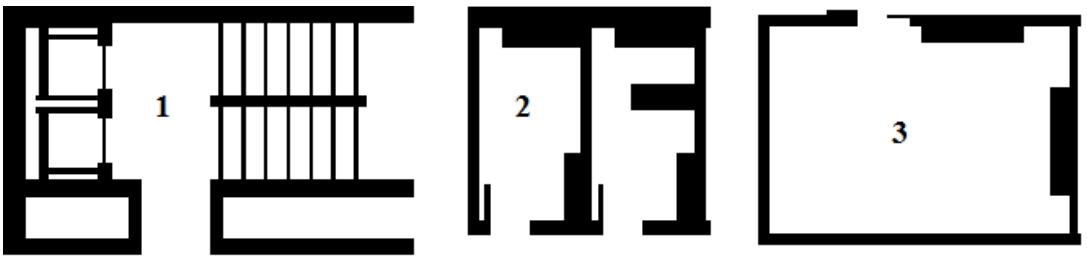


Figura 30 Rellano, despachos y laboratorio

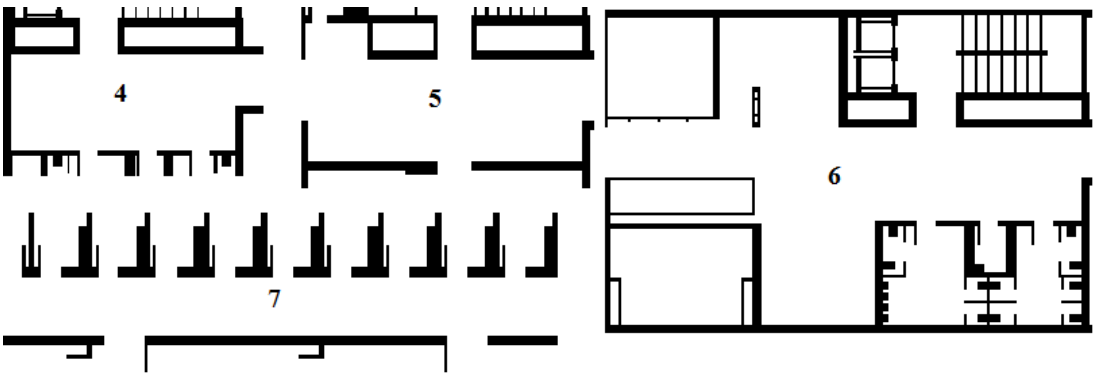


Figura 31 Pasillo y hall 1,2 y 3

Siendo respectivamente:

- Elemento 1: Rellano
- Elemento 2: Despacho
- Elemento 3: Laboratorio
- Elemento 4: Hall 1
- Elemento 5: Hall 2
- Elemento 6: Hall 3
- Elemento 7: Pasillo

El proceso de creación del modelo fue el siguiente.

En primer lugar se empleó la interfaz gráfica situándose el sensor en diferentes puntos de cada uno de los elementos y se guardaron los resultados de las lecturas. En función del detalle que buscamos para cada localización se emplearon más o menos lecturas, obteniendo de cada lectura alrededor de 60 puntos.

Los resultados se pueden guardar mediante la interfaz gráfica y se procesaron con Excel de manera individual particularizando las medidas para cada punto singular encontrado en las lecturas para posteriormente abrirlos con Matlab y emplearlos como datos de entrada de la librería Libsvm y crear los modelos.

El modelo generado es una variable sin representación gráfica, ya que esta ayuda visual no es estrictamente necesaria para resolver el problema de clasificación, pero con motivo de ilustrar el proyecto se empleó una función accesoria que permite el dibujo de los modelos (como en la figura 33) y facilita la comprensión de los mismos.

En cada uno de los elementos con los que se va a trabajar se localizaron puntos característicos que podrían ser identificados por el sensor y se desarrollaron tantos modelos como puntos hubiese, siendo dicho punto el origen de coordenadas del modelo.

Este método resulta costoso en tiempo ya que necesita crear un gran número de modelos que están referidos a todos los puntos característicos de la localización, pero después de realizar diversas pruebas para buscar una alternativa, no se encontró ninguna lo suficientemente válida que permitiera una clasificación robusta de las localizaciones con un coste computacional y de tiempo de trabajo bajos. Esta debería ser una línea de trabajo futuro sobre este proyecto, el reducir el número de modelos necesarios para comparar una localización y automatizar más su creación.

En el caso de la figura 32 (ejemplo de Hall 2), se decidieron como puntos de partida los bordes de puertas, finales de pared, esquinas, codos y finales de pasillo. Se representan con puntos rojos sobre la figura.

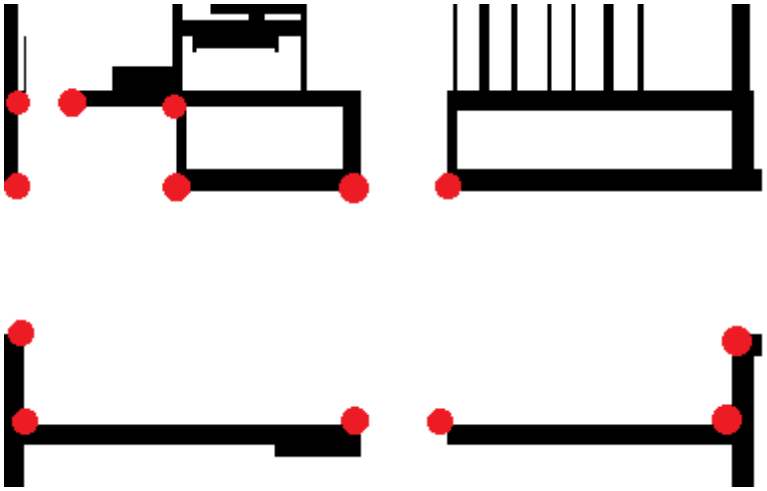


Figura 32 Puntos característicos en un Hall 2

Se observa que se requiere un número considerable de puntos singulares para representar localizaciones complejas. Una habitación cuadrada solo necesitará un punto por cada esquina y opcionalmente otros puntos en las puertas, mientras que un espacio como el hall representado necesita 13 sin ser exhaustivos. Conforme aumente la complejidad de la localización se necesitarán más puntos para aumentar la fiabilidad de la clasificación. Con pocos puntos, por ejemplo los cuatro que se emplearían en una habitación, es complicado acertar en la clasificación ya que distintas localizaciones pueden compartir parte de un modelo tomando como origen una esquina.

El número de datos etiquetados en cada modelo varía dependiendo del tamaño del mismo. Por término medio se emplean unos 250 puntos etiquetados como clase 1 (clase positiva, representados por una O en la figura 30) y otros 250 puntos como clase 0 o negativa.

El hiperplano separa la clase 1 del resto de puntos existentes, por lo que en teoría la clase 0 será todo lo que no sea clase 1, pero la toolbox de Matlab necesita para crear correctamente el modelo una serie de lecturas negativas que la ayuden a separar correctamente los datos.

En la figura 33 se observa el modelo del hall 2, como el contorno está formado por la clase 1 y se han generador suficientes puntos de clase 0 para ajustar el contorno. El modelo tiene suficiente detalle, pero se podría profundizar aún más, modelando los espacios situados detrás de cualquier puerta existente ya que podría darse el caso de que el robot enfoque directamente le hueco de una puerta desde el propio hall. Éste es un problema de eficiencia, siempre se puede mejorar el modelo, pero con el nivel de detalle que se maneja en este proyecto se llega a unos resultados bastante precisos. En el caso de que la clasificación no sea concluyente por estar enfocando una región no modelada completamente o por estar muy cerca de una pared o esquina (detalle que comparten muchos modelos) bastará con separar o girar el sensor para así cambiar la perspectiva de la medida.

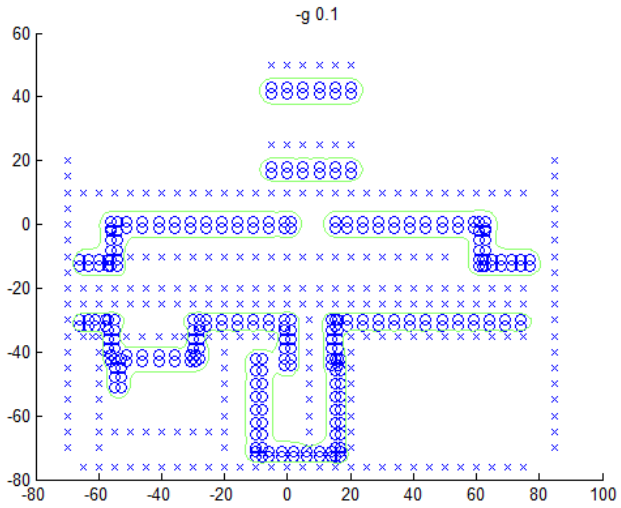


Figura 33 Modelo de un Hall 2

En total, para la correcta clasificación de las siete localizaciones se desarrollaron 135 modelos como se muestra en la siguiente tabla.

Algunas localizaciones más simples requirieron pocos modelos (como los pasillos), mientras que otras más complejas como algún hall, los laboratorios o los despachos necesitaron un número mayor de ellos para captar desde todos los ángulos los distintos elementos que las componen.

Tabla 7 Número de modelos por localización

Localización	Número de modelos
Pasillo	4
Hall 1	12
Hall 2	15
Hall 3	25
Laboratorio	37
Despacho	32
Rellano	10

Los pasillos, al estar formados por dos paredes paralelas no tienen esquinas ni codos. Para poder modelarlos se recrearon en el mapa las puertas con un ligero borde en el marco para que fuesen captadas por el sensor en caso de que estuviesen cerradas. Como las puertas son similares, solo son necesarios 4 puntos para modelar los pasillos.

Los tres tipos de hall necesitan un número mayor de modelos por los motivos expuestos sobre la figura 33. Se observa que el número de modelos es proporcional al tamaño de la localización, y al ser tercer hall el más grande también necesita más puntos característicos para simularlo.

Los 3 rellanos son muy parecidos, por lo que se empleó uno genérico para modelar los tres. Para esto se dibujaron las paredes más anchas de lo habitual ya que uno de los rellanos era más ancho que los otros dos.

Por último, en los laboratorios y despachos hay mucha variedad. Hay distintos laboratorios y despachos con distintos largos, anchos y bloques en las paredes, por lo que son necesarios más modelos que en las otras localizaciones para modelarlos todos.

Con estos modelos creados ya tenemos la base de datos contra la que realizar las clasificaciones con el robot.

4.4 CLASIFICACIÓN

El objetivo del proyecto es logar un resultado con suficiente fiabilidad en la clasificación de las siete localizaciones sobre el mapa de la universidad.

Una vez formada la base de datos con los modelos del apartado 4.3 y empleando la interfaz gráfica junto con la librería Libsvm se pueden simular distintos emplazamientos del sensor para comprobar la validez de este método de clasificación en problemas de localizaciones.

La función de SVM es llamada desde el código del proyecto una vez tenemos situado el robot y tomadas las medidas del sensor. Esta función de clasificación solo necesita un conjunto de datos de entradas en forma de vectores de dos posiciones espaciales (x e y), una etiqueta por cada par de datos y un modelo contra el que comparar los datos de entrada.

Una clasificación con SVM comprueba si los datos de entrada cuadran dentro del modelo y devuelve el resultado de la clasificación siendo éste positivo o negativo. La empleada en la librería Libsvm pide una etiqueta de entrada, por lo que al calcular el resultado lo compara con la predicción y determina si coinciden o no.

Esta utilidad se emplea en el proyecto de la siguiente manera. Se asigna una etiqueta positiva a todos los datos de entrada. Al clasificar dichos datos y compararlos con la etiqueta de entrada se obtiene la fiabilidad de la clasificación, ya que si todos los datos concuerdan con el modelo se obtendrá una coincidencia en las etiquetas del 100%, mientras que en caso contrario será del 0%.

El uso de la librería Libsvm se ampliará en el Anexo B.

En los próximos dos apartados se simularán clasificaciones con y sin ruido para probar la validez de las SVM.

4.5 RESULTADOS SIN RUIDO

Para probar la validez del planteamiento se comenzará con clasificaciones de localizaciones sin ruido. Para ello se empleará la herramienta descrita en el apartado 4.1 con el mapa que viene por defecto (con las puertas abiertas). Se simularán 40 medidas en cada ubicación (el listado aparece en el Anexo A del presente proyecto).

Las simulaciones se harán enfocando con el sensor hacia todas las direcciones posibles y a distintas distancias de las paredes y esquinas para crear una muestra de medidas representativas.

En el caso de zonas tipo rellano, despacho o laboratorio, de las que hay varias en el mapa, se realizarán medidas en todas por igual.

En la siguiente figura se representan parte de las medidas del Hall 1 (el vértice del triángulo representa la dirección del sensor). En rojo los puntos en los que se clasificó correctamente como Hall 1 y el azul los puntos no clasificados.

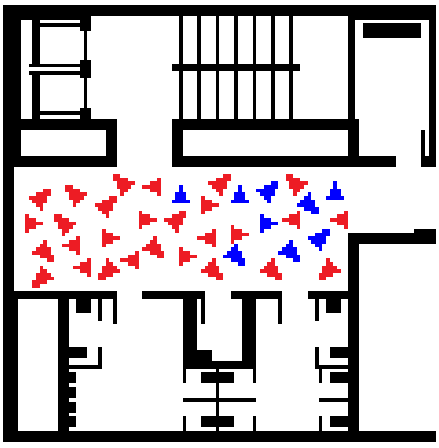


Figura 34 Resultados de la clasificación en el Hall 1

En este caso en particular, como se comentará más adelante, se dan clasificaciones erróneas en las zonas en que el hall se confunde con el pasillo. Asimismo, dada la similitud entre el hall 1 y zonas del hall 3 se produce confusión entre ambas localizaciones.

Los resultados completos de las siete localizaciones se muestran a continuación.

En la primera tabla las clasificaciones aparecen en valor absoluto: en negrita los aciertos y en rojo fallos.

La segunda tabla reproduce los resultados de la primera pero en porcentaje sobre las 40 medidas.

Tabla 8 Resultado de la clasificación en valor absoluto

	Clasificado como:							
	Sin coincidencias	Pasillo	Hall 1	Hall 2	Hall 3	Laboratorio	Despacho	Rellano
Pasillo	1	**39**	0	0	0	0	0	0
Hall 1	0	2	**31**	0	7	0	0	0
Hall 2	0	1	0	**36**	3	0	0	0
Hall 3	0	1	4	0	**35**	0	0	0
Laboratorio	0	0	0	0	1	**39**	0	0
Despacho	0	0	0	0	0	3	**37**	0
Rellano	0	0	0	0	2	5	0	**33**

En la siguiente tabla se muestra el resultado en %

Tabla 9 Resultado de la clasificación en porcentaje

	Clasificado como (en %):							
	Sin coincidencias	Pasillo	Hall 1	Hall 2	Hall 3	Laboratorio	Despacho	Rellano
Pasillo	2,5	**97,5**	0	0	0	0	0	0
Hall 1	0	5	**77,5**	0	17,5	0	0	0
Hall 2	0	2,5	0	**90**	7,5	0	0	0
Hall 3	0	2,5	10	0	**87,5**	0	0	0
Laboratorio	0	0	0	0	2,5	**97,5**	0	0
Despacho	0	0	0	0	0	7,5	**92,5**	0
Rellano	0	0	0	0	5	12,5	0	**82,5**

Los resultados son esperanzadores pero no del todo satisfactorios. Hay localizaciones que se clasifican correctamente en un % muy alto, mientras que otras tienen errores superiores al 20%.

Pasillo:

Se clasifican correctamente como pasillo el 97,5 % de las observaciones (39/40). Es una zona que no guarda similitud con ninguna otra por lo que el acierto es alto.

Se dan clasificaciones erróneas de otras zonas como pasillo debido a que si el sensor se sitúa frente a una pared de un Hall, el resultado es prácticamente el mismo a estar frente a una pared de un pasillo.

Hall 1:

Se clasifican como Hall 1 solo el 77,5% de las medidas (31/40). Se produce confusión con los espacios Hall 3 y Pasillo.

Los datos clasificados como Pasillo se deben a que la medida se tomó en una zona de transición. Este hecho resulta muy complicado de resolver de manera estática. Sin embargo de manera dinámica, esto es, con el sensor en movimiento se partiría de una situación en la que se reconocería con una fiabilidad alta la localización como Hall 1 y se iría reduciendo dicha fiabilidad conforme se acercase a la zona de transición.

Los clasificados como Hall 3 se deben a la similitud entre ambas localizaciones, un hecho fortuito que es difícil de evitar.

Hall 2:

El acierto llega al 90% produciéndose confusión con modelos del Hall 3 por el mismo motivo que ocurría en el Hall 1.

Hall 3:

Se clasifican correctamente 35/40. Las muestras mal clasificadas se deben al mismo motivo que las del Hall 1, ambos modelos comparten zonas idénticas por lo que el programa es incapaz de diferenciar entre ellas.

Laboratorio:

Alto acierto de clasificación (97,5%). Son zonas amplias y bien diferenciadas del resto.

Despacho:

Se clasifican como despacho 37 de 40 muestras. Las confusiones vienen dadas cuando el sensor enfoca directamente una esquina, ya que sin más información del entorno no se diferencia de una esquina de laboratorio.

Rellano:

Los errores de clasificación en la ubicación ‘rellano’ son los mismos que en los despachos. Cuando la lectura del sensor es de una esquina, sin aportar más detalles significativos, al programa le resulta imposible diferenciarlo de un laboratorio.

4.5.1 PROBLEMAS

Las simulaciones de cada punto del sensor resultan bastante costosas en tiempo y computación. Clasificar una serie de medidas contra un único modelo es instantáneo, pero se complica cuando se tienen que cruzar 130 modelos. Al emplear una clasificación 1-v-r tenemos que llamar 130 veces a la función de SVM por cada punto característico hallado por el sensor, lo que pueden ser entre 500 y 1000 veces. Ya que en cada medida se encuentran varios puntos (desde 2 o 3 en una lectura de un espacio simple, a 10 o más en lecturas en espacios complejos, con salientes y obstáculos).

Con la interfaz y las funciones empleadas en este proyecto la primera simulación realizada después de abrir el programa por primera vez dura entorno a los 10 segundos, siendo la duración de las siguientes inferior al segundo. Esta diferencia de tiempos puede deberse a que Matlab carga en memoria datos la primera vez que se ejecuta.

El tiempo de primera ejecución no resulta un problema demasiado grave, ya que solo afecta a la primera simulación, siguiendo una analogía electro-mecánica, sería el transitorio del problema, no afectando al régimen permanente.

El segundo que tarda en clasificar una medida ‘en régimen permanente’ no resulta muy grave para la resolución de problemas estáticos del sensor pero si queremos aplicar esta modelización a un robot real, un segundo es demasiado tiempo ya que el robot comprobaría instantáneamente donde se encuentra y requeriría de un método más rápido.

4.6 RESULTADOS CON RUIDO

Los resultados obtenidos en simulaciones sin ruido, como se comentó en el apartado 4.5, son aceptables, pero en situaciones reales encontraremos cierta cantidad de ruido por lo que es necesario comprobar la validez del planteamiento con obstáculos que interfieran la medida del sensor.

Sobre la interfaz hay un botón en el cuadro 'clasificación' nombrado como 'Obstáculos'. Mediante este botón se carga un mapa con ruido en las localizaciones objeto de estudio. Sobre este mapa se harán las medidas del sensor.

Simulamos 15 medidas en cada localización con ruido, representado en forma de obstáculos, para comprobar la validez de la clasificación con SVM de datos tipo localización en dos dimensiones en entornos reales.

En las figuras 35 y 36 se representan las 7 localizaciones con ruido (color rojo).

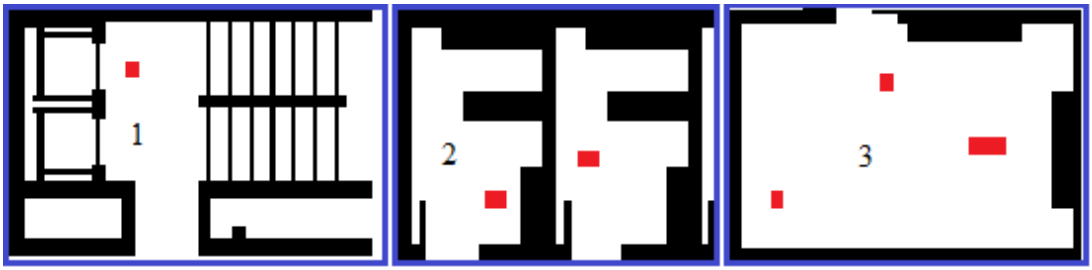


Figura 35 Rellano, despacho y laboratorio con ruido

Figura 36 Pasillo y hall 1,2 y 3 con ruido

Se siguió el mismo sistema de simulación que para el mapa sin ruido. Las simulaciones se hacen enfocando hacia todas las direcciones posibles y a distintas distancias de las paredes y esquinas para crear una muestra de medidas representativas.

En las zonas tipo rellano, despacho o laboratorio, de las que hay varias en el mapa, se realizarán medidas en todas por igual.

El resultado completo de las simulaciones se encuentra en el anexo.

En la tabla 10 se muestran las conclusiones de la simulación. En la columna 'coincidencias' aparece el número de veces, tanto en valor absoluto como en tanto por ciento, que la clasificación de los datos con y sin ruido fue la misma. En la de 'pérdida de fiabilidad media' aparece la diferencia media en % entre la fiabilidad de la clasificación de los resultados con y sin ruido, siempre que haya habido coincidencia en la clasificación.

Tabla 10 Resultado de las simulaciones con ruido

Localización	**Nº medidas**	**Coincidencias**	**Pérdida de fiabilidad media**
Pasillo	15	15 (100 %)	2,594 %
Hall 1	15	7 (46,67 %)	7,888 %
Hall 2	15	12 (80 %)	7,060 %
Hall 3	15	13 (86,67 %)	6,615 %
Laboratorios	15	13 (86,67 %)	6,442 %
Despacho	15	15 (100 %)	4,020 %
Rellano	15	12 (80 %)	5,853 %

Los resultados son favorables para los modelos que ya dieron buen resultado en el ensayo sin ruido como son despacho y pasillo. Estas localizaciones tienen perfiles bien diferenciados y el hecho de que existan obstáculos solo reduce proporcionalmente la fiabilidad en la medida, pero no hace que se confundan modelos.

En otras localizaciones como en los rellanos o laboratorios se producen fallos de coincidencia, esto es, que en vez de clasificar una medida como 'laboratorio' (siendo tomada en un laboratorio) la clasifica como otra localización. Esto se debe a que determinados puntos característicos se tapan con los obstáculos o que el mismo obstáculos muestra puntos característicos que son similares los de otra localización. También se debe a que no encuentra coincidencias ya que los obstáculos reducen la fiabilidad de la clasificación por debajo del límite elegido en la interfaz y eso provoca que la función no devuelva una clasificación válida.

En localizaciones como los halls la existencia de obstáculos aumenta la pérdida de fiabilidad en las medidas lo que provoca que se pasen del límite y clasifiquen erróneamente la localización. Esto se hace patente en el hall 1 que era la localización con más errores en las pruebas sin ruido y en el momento en que introducimos obstáculos.

El mayor de los problemas al tratar con obstáculos es el expuesto al final del capítulo 3, toda medida tomada en la proximidad de un obstáculo sale muy distorsionada.

Esto es un problema menor a la hora de trabajar con un sensor en movimiento porque se percibiría que el error de la clasificación va aumentando conforme nos acercamos a un obstáculo y el robot podría realimentar esa situación y separarse, comparando siempre la fiabilidad en la clasificación para deducir obstáculos y zonas de transición.

CAPITULO 5. CONCLUSIONES

5.1 CONCLUSIONES RESPECTO A LOS RESULTADOS

A los largo de este proyecto se han expuesto los principios de funcionamiento de los sistemas de clasificación de datos basados en máquinas de soporte vectorial. Además se demostró su utilidad para clasificar datos recibidos de un sensor que capte medidas de localizaciones en dos dimensiones y su aplicación a un caso concreto de reconocimiento y clasificación de 7 ubicaciones conocidas sobre un mapa de la Universidad Carlos III.

En cuanto a la fiabilidad de los resultados obtenidos tenemos que la clasificación en espacios cerrados o espacios simples es muy buena, 97,5% para pasillos o laboratorios y del 92,5% para despachos. Mientras que para clasificar espacios abiertos o con finales/salidas menos definidos la fiabilidad baja mucho ya que en las zonas de transición es muy complicado distinguir donde empieza una zona y donde termina la anterior ya que las medidas se dividen en dos mitades entre las zonas.

Los resultados con ruido son los esperados, reduciendo la fiabilidad a la hora de clasificar. En las localizaciones cerradas perdemos fiabilidad pero mantenemos una buena clasificación, mientras que en las abiertas o poco definidas los resultados no son satisfactorios.

En cuanto a los tiempos de simulación, la clasificación 1-v-r no es la adecuada por los elevados tiempos de cálculo, ya que requiere el empleo de muchos modelos y de muchas llamadas a la función clasificadora de SVM.

Con estas consideraciones (de coste y resultado) podemos afirmar que la clasificación 1-v-r de SVM para reconocer localizaciones de manera genérica no es la mejor. Puede ser válida para reconocer entornos que sean muy distintos entre sí, pero a la hora de diferenciar entre patrones similares o que no tengan muy definidos sus contornos no da buenos resultados.

5.2 TRABAJO FUTURO

Como los resultados no los deseados se podría mejorar la programación existente en varias vertientes:

- Mejorar la clasificación manteniendo el tipo de SVM empleada (C-SVM tipo clasificación) variando los valores del kernel. En este proyecto se decidió emplear un tipo de kernel común a todos los modelos, pero podría darse la

situación de que para distintos modelos encontremos mejores resultados empleando distintos kernel. Asimismo, empleando otros kernel y variando los parámetros gamma, C y NU se puede llegar a resultados con menor coste computacional (por ejemplo en kernels lineales) e igual fiabilidad.

- Emplear otro tipo de SVM. La máquina de soporte vectorial de estimación-distribución (SVM de una clase) tiene la ventaja de trabajar con una única clase, la localización que deseamos modelar.
- Reorientar la programación para reducir el número de clasificaciones y el coste computacional. En la actual dinámica de trabajo el número de modelos está sobredimensionado y eso produce clasificaciones muy lentas.

También sería interesante comparar los resultados que se obtendrían con un algoritmo de tipo 1-v-1 (one-versus-one). En éste se compara cada clase K con cada una de las restantes, lo que supone realizar K(K-1)/2 clasificaciones. Realizaríamos más clasificaciones lo que a priori supone mayores tiempos de cálculo, por eso empleamos en el presente proyecto el algoritmo 1-v-r, pero no sabemos si arrojaría mejores resultados.

5.3 OTRAS APLICACIONES

Basándonos en el desarrollo del proyecto, las SVM con un sistema de clasificación 1-v-r son un buen algoritmo para otro tipo de trabajos, como el reconocimiento de escritura y números. Un posible campo de aplicación sería el reconocimiento automático de las lecturas de los contadores de agua, electricidad o gas de las viviendas mediante una fotografía de los mismos.

En los casos en que los datos proporcionados al algoritmo de SVM estén en la misma escala, solo será necesario un modelo por ‘objeto’ lo cual simplifica mucho el procedimiento.

En el presente proyecto cada lectura era distinta de la anterior por lo que había que buscar puntos comunes para referir las medidas a los modelos creados, por lo que el número total de modelos fue excesivamente alto (130), frente a los 10 que habría que crear para modelar cualquier número existente, ya que cualquier número se puede separar en cifras y solo hay 10 cifras. Algo similar a lo que ocurriría en el reconocimiento de textos ya que habría que crear 26 modelos, uno por cada letra (si excluimos letras como la ñ o la cedilla).

Otros campos de aplicación distintos en los que se trabaja con SVM son el reconocimiento de sonidos y voces, mediante el análisis de frecuencias. Se podría plantear resolver el reconocimiento de un avión despegando por el ruido que hace su motor o distinguir entre diferentes cantantes por la clasificación de sus canciones cuando suenen en la radio.

CAPITULO 6. BIBLIOGRAFÍA

6.1 LIBROS, ARTÍCULOS Y PÁGINAS WEB

(1) JALON J.C., J.I. RODRIGUEZ, J. VIDAL. *Aprenda Matlab 7.0 como si estuviera en primero*. Universidad Politécnica de Madrid, 2005.

(2) *LIBSVM. A library for Support Vector Machines* http://www.csie.ntu.edu.tw/~cjlin/libsvm/

(3) BORRAJO, Daniel. *Apuntes de la asignatura Inteligencia Artificial curso 2008-2009*. Departamento de informática, Universidad Carlos III

(4) *Asociación española de inteligencia artificial (AEPIA)*. http://www.aepia.org/

(5) PUELLES, Luis, SÁNCHEZ CÁNOVAS, José, ALBERTOS, Pedro. *Inteligencia artificial e inteligencia humana*.

(6) *Inteligencia Artificial y temas relacionados. Wikipedia*. http://es.wikipedia.org/wiki/Inteligencia_artificial

(7) SÁNCHEZ G., PÉREZ H., NAKANO M. *Growing Cell Neural Network using Simultaneous Perturbation*. Inst. Politécnico Nacional, Secc. de Estudios de Posgrado e Investigación, Esc. Superior de Ingeniería Mecánica y y Eléctrica. México D. F.

(8) SÁNCHEZ-MONTAÑÉS ISLA, Manuel A. *Métodos Avanzados en Aprendizaje Artificial. Redes neuronales.* Universidad Autónoma de Madrid, 2006

(9) *History of the perceptron*. http://www.csulb.edu/~cwallis/artificialn/History.htm

(10) *Perceptrón. Wikipedia* http://es.wikipedia.org/wiki/Perceptr%C3%B3n

(11) *Multilayer perceptrons*. http://users.ics.tkk.fi/ahonkela/dippa/node41.html

(12) *Perceptrón multicapa. Wikipedia* http://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa

(13) YU JIANGSHENG. *Method of k-Nearest Neighbors*. Institute of Computational Linguistics Peking University, China, 2002

(14) DE LA ESCALERA, Arturo, ARMINGOL, José maría. *Apuntes de la asignatura Sistemas de percepción* (proveniente del libro Visión por Computador,

fundamentos y métodos, Arturo de la Escalera Hueso. Prentice Hall). Departamentos de ingeniería de sistemas y automática, Universidad Carlos III

(15) FERNÁNDEZ REBOLLO, Fernando, BORRAJO MILLÁN, Daniel, GARCÍA DURÁN, Rocío. *Apuntes de la asignatura Aprendizaje automática*. Departamento de informática, Universidad Carlos III

(16) CHIH-WEI HSU, CHIH-CHUNG CHANG AND CHIH-JEN LIN. *A Practical Guide to Support Vector Classification*

(17) CHISTOPHER J.C. BURGES. *A Tutorial on Support Vector Machines for Pattern Recognition* (Appeared in: Data Mining and Knowledge Discovery 2, 121-167, 1998)

CAPITULO 7. ANEXOS

7.1 ANEXO A. RESULTADOS

En el primer anexo del proyecto se muestran los resultados completos tabulados de las simulaciones de prueba post-entrenamiento sin obstáculos y con obstáculos (ruído).

7.1.1 RESULTADOS SIMULACIÓN SIN OBSTÁCULOS

En el presente apartado se adjuntan los resultados de la simulación del programa para las localizaciones:

- Pasillo
- Hall 1
- Hall 2
- Hall 3
- Laboratorio
- Despacho
- Rellano

Como se comentó en el capítulo 4, el programa localiza puntos característicos mediante el método de las distancias euclídeas y tomando dichos puntos como referencia de origen comparaba las medidas adquiridas por el sensor con los modelos desarrollados de las superficies.

En las siguientes tablas se muestran 40 simulaciones por cada localización con el resultado de la clasificación, su fiabilidad en % (número de medidas clasificadas como resultado / número de medidas totales) y la subclasificación de cada punto característico.

Las abreviaturas de las tablas son:

- Fiab (%): fiabilidad de la clasificación
- SC: Sin clasificar
- PAS: Pasillo
- LAB: Laboratorio
- RELL: Rellano
- DES: Despacho

- H1: Hall 1
- H2: Hall 2
- H3: Hall 3

A modo de ejemplo se explica la segunda medida tomada en la ubicación 'pasillo'.

	Resultado	Fiab (%)	SC	PAS	LAB	RELL	DES	H1	H2	H3
2	Pasillo	92,9412	1	5						

Situado el sensor en una zona de pasillo el modelo informático encuentra 6 puntos característicos. Uno de ellos no coincide con suficiente fiabilidad (tenemos definido el mínimo en un 80%) con ningún modelo. Los otros 5 coinciden con un modelo de pasillo.
La fiabilidad global de la clasificación es del 92,9412 %, esto es, 93 de cada 100 medidas tomadas por el sensor se etiquetan como pasillo. Las medidas que no son etiquetadas correctamente suelen deberse a las coincidentes con puertas abiertas o finales de pasillo.

Debido a esto el programa clasifica la observación como un pasillo.

Pasillo

Tabla 11 Resultados simulación pasillo sin ruido

	Resultado	Fiab (%)	SC	PAS	LAB	RELL	DES	H1	H2	H3
1	Pasillo	100		6						
2	Pasillo	92,9412	1	5						
3	Pasillo	82,619	2	5						
4	Sin coincidencias	0								
5	Pasillo	95,2941		3			1			
6	Pasillo	90,6977		3						1
7	Pasillo	96,4706		3						1
8	Pasillo	94,1176	1	3			1			
9	Pasillo	100		3						
10	Pasillo	100		3						
11	Pasillo	100		4						
12	Pasillo	91,7647		4						
13	Pasillo	92,9412		2						
14	Pasillo	100		2						
15	Pasillo	100	1	3						
16	Pasillo	100		3						
17	Pasillo	87,8431	2	3						
18	Pasillo	95,5926		3						1

19	Pasillo	95,2381	1	4						
20	Pasillo	91,6667		4						
21	Pasillo	98,8095		5				1		
22	Pasillo	82,3529	2	3						
23	Pasillo	95,2941	1	2						
24	Pasillo	98,8235		3				1		
25	Pasillo	91,9643		4						
26	Pasillo	82,1429	1	3				1		
27	Pasillo	100	1	2						
28	Pasillo	89,0196	1	3						
29	Pasillo	82,0588	2	4						
30	Pasillo	96,4286		2						1
31	Pasillo	99,4048		2						
32	Pasillo	97,6471	3	4						
33	Pasillo	93,75	2	4						
34	Pasillo	98,1928	2	4						
35	Pasillo	99,7647	1	5						
36	Pasillo	100		2						
37	Pasillo	100	1	2						
38	Pasillo	100		2						
39	Pasillo	100		2						
40	Pasillo	96,4286	1	4						

Hall 1

Tabla 12 Resultados simulación hall 1 sin ruido

	Resultado	Fiab (%)	SC	PAS	LAB	RELL	DES	H1	H2	H3
1	Hall 1	92,6471	2	0	0	0	0	4	0	0
2	Hall 1	95,2941	2	0	0	0	0	3	0	0
3	Hall 1	90,5882	2					4		1
4	Hall 1	96,1765	1					4		1
5	Hall 3	83,0588	2							5
6	Hall 3	91,7647		2						3
7	Hall 1	96,2963						3		1
8	Hall 1	98,8889						4		1
9	Hall 1	88,8889	2					4		1
10	Hall 1	96,6667	1					4		1
11	Hall 1	92,2222	2					6		1

12	Pasillo	86,6667	1	7				1		
13	Hall 1	97,7778	1	1				4		1
14	Hall 1	96,2222	2					5		1
15	Hall 1	100						2		1
16	Hall 1	100	1					4		1
17	Hall 1	100	1					4		1
18	Hall 1	100	1					5		
19	Hall 1	100	0					2		
20	Hall 1	100	1					2		
21	Hall 1	100	1					3		
22	Hall 1	100						3		
23	Hall 1	100						2		
24	Hall 3	96,5116	2	1						3
25	Pasillo	96,0784		3						
26	Hall 3	90,9804	2							3
27	Hall 3	100	1							4
28	Hall 1	100	1					5		
29	Hall 1	95,2381	2					3		1
30	Hall 1	92,9412	3					5		
31	Hall 3	96,0317	4							3
32	Hall 3	95,2941	1							3
33	Hall 1	96,9444						4		3
34	Hall 1	96,9841	2					7		1
35	Hall 1	93,5556	1					5		
36	Hall 1	95,5556	3					5		
37	Hall 1	100	1					2		
38	Hall 1	87,7778	1	1				6		
39	Hall 1	95,9259	2					3		1
40	Hall 1	88,0556	3	1				4		

Hall 2

Tabla 13 Resultados simulación hall 2 sin ruido

	Resultado	Fiab (%)	SC	PAS	LAB	RELL	DES	H1	H2	H3
1	Hall 2	90,1235	2						3	
2	Hall 2	95							3	
3	Hall 2	96,2963	1						6	
4	Hall 2	100							2	

5	Hall 2	96,4706	2						4	
6	Hall 2	96,4706							3	
7	Hall 3	97,6744	2						1	2
8	Hall 2	92,9412	1						3	
9	Hall 3	90,4762	0							1
10	Hall 2	100						1	2	
11	Hall 2	100		1					1	
12	Hall 2	100	1						3	
13	Hall 2	95,8333	1						2	
14	Hall 2	97,6471	1						3	1
15	Hall 2	96,4706	3						2	
16	Hall 2	95,3488	2						4	
17	Hall 2	96,4706	3						3	
18	Hall 2	97,6744	2						5	1
19	Hall 2	91,7647	2						4	
20	Hall 2	87,8431	2						6	
21	Hall 2	87,9518	1	1					2	
22	Hall 2	92,5926	1	1					2	
23	Hall 2	97,6471	3						1	
24	Hall 2	100							2	
25	Hall 2	99,1667							3	
26	Hall 2	88,2353	2	1					3	
27	Hall 2	98,5944		1					6	
28	Hall 3	98,4314		1						3
29	Hall 2	95,6349		1					3	
30	Pasillo	99,3976		2						
31	Hall 2	96,1765	2						4	
32	Hall 2	91,4683	1	1					6	
33	Hall 2	93,1973	2						7	
34	Hall 2	100	1						3	
35	Hall 2	100		1					2	
36	Hall 2	94,1176	3						3	2
37	Hall 2	100		2						
38	Hall 2	93,1548	1						4	
39	Hall 2	99,4318	2						2	
40	Hall 2	93,0041	2						3	

Hall 3

Tabla 14 Resultados simulación hall 3 sin ruido

	Resultado	Fiab (%)	SC	PAS	LAB	RELL	DES	H1	H2	H3
1	Pasillo	100		4						
2	Hall 3	100	2							3
3	Hall 3	100	1							3
4	Hall 3	100	3							4
5	Hall 3	100						1		3
6	Hall 3	93,5185	1							4
7	Hall 3	100								2
8	Hall 3	100								3
9	Hall 3	98,8764								3
10	Hall 3	97,037	2							3
11	Hall 3	82,963	1							6
12	Hall 3	88,2222	2							5
13	Hall 3	84,5783	2							5
14	Hall 3	91,1111								5
15	Hall 3	100								2
16	Hall 3	100			1					1
17	Hall 3	100								3
18	Hall 3	98,2353	1							4
19	Hall 3	82,5397	1							7
20	Hall 3	95,5294	2							5
21	Hall 3	86,4286	1							7
22	Hall 3	81,2048	2							5
23	Hall 3	100								3
24	Hall 3	88,9412	2							5
25	Hall 3	76,4706	2							5
26	Hall 3	94,8413	2							6
27	Hall 3	91,0714	2							6
28	Hall 1	86,6667	1					4		2
29	Hall 3	96,7059	1							5
30	Hall 1	98,7179						3		1
31	Hall 1	100								
32	Hall 3	81,9405	2							4
33	Hall 3	84,0476	4							5
34	Hall 1	96	1					4		2
35	Hall 3	93,3908	2							4
36	Hall 3	81,4118	2							4
37	Hall 3	100	1							5
38	Hall 3	82,1176	2							5
39	Hall 3	82,5893								8

40	Hall 3	100								3

Laboratorio

Tabla 15 Resultados simulación laboratorio sin ruido

	Resultado	Fiab (%)	SC	PAS	LAB	RELL	DES	H1	H2	H3
1	Laboratorio	100			1					
2	Laboratorio	96,2963			3					
3	Laboratorio	100			4					1
4	Laboratorio	92,2222	1		5					
5	Laboratorio	100			2					
6	Laboratorio	92,2222			3					
7	Laboratorio	92,4074	1		6					
8	Laboratorio	100	1		2					
9	Laboratorio	100			2					
10	Laboratorio	93,3333	2		6					
11	Laboratorio	100	2		3		1			
12	Laboratorio	100			3					1
13	Laboratorio	100			3					
14	Laboratorio	100			2					
15	Laboratorio	97,7778			2					
16	Laboratorio	92,5397	1		7					
17	Laboratorio	92,2222	2		5					
18	Laboratorio	94,8148	1		3					
19	Hall 3	91,1111								2
20	Laboratorio	94,4444	1		3					
21	Laboratorio	96,6667	1		3					
22	Laboratorio	88,8889	2		2					
23	Laboratorio	95,5556			1					1
24	Laboratorio	94,4444			2					
25	Laboratorio	87,7778	2		2					
26	Laboratorio	95,5556			3					
27	Laboratorio	100			4					
28	Laboratorio	91,1111	1		6					
29	Laboratorio	95,5056	1		3					
30	Laboratorio	92,2222			4					
31	Laboratorio	89,7778	2		5					
32	Laboratorio	94,8148			3					

33	Laboratorio	100			3					
34	Laboratorio	100			2					
35	Laboratorio	98,3333			4		1			
36	Laboratorio	100			3					
37	Laboratorio	93,2584			3					
38	Laboratorio	100			1					
39	Laboratorio	100			4					
40	Laboratorio	94,1667		1	4					

Despacho

Tabla 16 Resultados simulación despacho sin ruido

Resultado	Fiab (%)	SC	PAS	LAB	RELL	DES	H1	H2	H3
1	Despacho	100					3			
2	Despacho	100					2			
3	Despacho	100		1			2			
4	Despacho	94,0741					3			2
5	Despacho	99,2593	1				3			
6	Despacho	97,4074	1				3			
7	Despacho	98,0556			1		4			1
8	Despacho	100					2			
9	Despacho	90,2778	1				4			
10	Despacho	95,8333	1				4			
11	Despacho	94,0741					3			
12	Despacho	93,5556					5			
13	Despacho	100					4			1
14	Laboratorio	100		1	1					
15	Laboratorio	98,8889		1	1		4			
16	Despacho	99,4444	1		2		2			
17	Despacho	100					3			
18	Despacho	100			1		3			
19	Despacho	95,5556	1				4			
20	Laboratorio	100		1	1					
21	Despacho	100					3			2
22	Despacho	91,6667	1				6			
23	Despacho	96,6667					4			
24	Despacho	96,6667					5			
25	Despacho	100					2			1

26	Despacho	97,7778	2				4			
27	Despacho	92,2222	1				2			1
28	Despacho	99,6296	2				3			
29	Despacho	92,4444	1				5			
30	Despacho	95,6818					5			
31	Despacho	93,3333	1				3			
32	Despacho	96,2963	1				6			
33	Despacho	97,037					3			
34	Despacho	97,7778					3			1
35	Despacho	92,7778	1				4			
36	Despacho	94	1				5			
37	Despacho	95,9259					3			1
38	Despacho	97,037	2				3			
39	Despacho	95,1111					5			
40	Despacho	95,7386					4			

Rellano

Tabla 17 Resultados simulación rellano sin ruido

	Resultado	Fiab (%)	SC	PAS	LAB	RELL	DES	H1	H2	H3
1	Rellano	100	1	1		3				
2	Laboratorio	100		3						
3	Laboratorio	100		2	1	1				1
4	Rellano	96,6667				4				
5	Rellano	85,2273				4		2		1
6	Rellano	92,0455	2	1		5				1
7	Rellano	100				3				
8	Rellano	98,1481	1			6				
9	Rellano	90,9091	1			3		2		2
10	Rellano	98,2222	1			5				1
11	Laboratorio	100			4					
12	Rellano	87,1111	2			5		1		
13	Rellano	95,8333	1			4				
14	Rellano	97,4074				3				
15	Rellano	99,2593	3			3				
16	Rellano	96,6667	1	1		5				
17	Rellano	100	1		1	2				
18	Rellano	100		1		3				1

19	Rellano	100	1		1	2	1			
20	Laboratorio	97,7778				3			1	1
21	Hall 3	90,8046				3			2	3
22	Rellano	88,5057			2	3			2	1
23	Rellano	88,764	1			3	1		1	
24	Rellano	85,5556	2			2				1
25	Rellano	96,3889			1	4				
26	Hall 3	96,1111				1	2		1	2
27	Rellano	95,5556				3		1		
28	Rellano	98	1			5				
29	Rellano	100	1		1	4				
30	Rellano	98,8889				5				1
31	Rellano	100				3				2
32	Rellano	93,6111				4				
33	Rellano	98,8889		1		4				
34	Laboratorio	100			1					
35	Rellano	100		1		3			1	
36	Rellano	100		1		4			1	
37	Rellano	100	3			3				
38	Rellano	94,4444				2	1	1		1
39	Rellano	90,1149				5		2		1
40	Rellano	89,2593				3				1

7.1.2 RESULTADOS COMPARACIÓN SIMULACIÓN CON Y SIN OBSTÁCULOS

Se muestran las simulaciones, 15 por cada tipo de espacio, primero sin obstáculos y a continuación con obstáculos para comparar como influye el 'ruido' en la clasificación.

Los resultados fueron analizados en el capítulo 5.

Las abreviaturas correspondientes a las tablas son:

- Resultado SO: resultado de la clasificación sin obstáculos
- Fiab (%): fiabilidad de la clasificación
- Resultado CO: resultado de la clasificación con obstáculos
- Comp: Comparación entre el resultado sin y con obstáculos. SI, si coinciden, NO si difieren
- Dif(%): Diferencia en % entre la fiabilidad en la clasificación sin obstáculos y con obstáculos

Pasillo

Tabla 18 Resultados simulación pasillo con ruido

	Resultado SO	Fiab (%)	Resultado CO	Fiab (%)	Comp.	Dif. (%)
1	Pasillo	98,8235	Pasillo	95,6395	SI	3,184
2	Pasillo	100	Pasillo	100	SI	0
3	Pasillo	100	Pasillo	97,7011	SI	2,2989
4	Pasillo	100	Pasillo	100	SI	0
5	Pasillo	96,4286	Pasillo	95,5814	SI	0,8472
6	Pasillo	98,8235	Pasillo	92,7203	SI	6,1032
7	Pasillo	86,9048	Pasillo	85,0575	SI	1,8473
8	Pasillo	100	Pasillo	90,1149	SI	9,8851
9	Pasillo	84,4706	Pasillo	82,5581	SI	1,9125
10	Pasillo	82,5581	Pasillo	82,5581	SI	0
11	Pasillo	98,8095	Pasillo	97,2868	SI	1,5227
12	Pasillo	100	Pasillo	99,0698	SI	0,9302
13	Pasillo	100	Pasillo	90,8046	SI	9,1954
14	Pasillo	97,6471	Pasillo	96,4563	SI	1,1908
15	Pasillo	100	Pasillo	100	SI	0

Hall 1

Tabla 19 Resultados simulación hall 1 con ruido

	Resultado SO	Fiab (%)	Resultado CO	Fiab (%)	Comp.	Diferencia
1	Hall 1	96,1765	Hall 3	87,2222	NO	8,9543
2	Hall 1	95,0588	Hall 1	90,5882	SI	4,4706
3	Hall 1	96,4706	Hall 1	83,9216	SI	12,549
4	Hall 1	87,5556	Pasillo	85,5556	NO	2
5	Hall 1	91,4815	Sin coincid.	0	NO	91,4815
6	Hall 1	93,1481	Sin coincid.	0	NO	93,1481
7	Hall 1	95	Hall 1	86,2069	SI	8,7931
8	Hall 1	100	Hall 1	88,8889	SI	11,1111
9	Hall 3	94,5736	Hall 3	88,5057	SI	6,0679
10	Hall 1	91,1111	Pasillo	87,2222	NO	3,8889
11	Hall 1	97,4074	Hall 3	87,2222	NO	10,1852
12	Hall 1	92,2222	Hall 1	85,5556	SI	6,6666

13	Hall 1	100	Hall 3	81,111	NO	18,889
14	Hall 1	94,4444	Hall 1	88,8889	SI	5,5555
15	Hall 1	87,7778	Pasillo	85,7778	NO	2

Hall 2

Tabla 20 Resultados simulación hall 2 con ruido

	Resultado SO	Fiab (%)	Resultado CO	Fiab (%)	Comp.	Diferencia
1	Hall 2	97,1875	Hall 2	85,3125	SI	11,875
2	Hall 2	88,8889	Hall 2	86,2198	SI	2,6691
3	Hall 2	89,8148	Hall 2	87,8049	SI	2,0099
4	Hall 2	98,7952	Hall 2	84,1176	SI	14,6776
5	Hall 2	94,6218	Hall 2	89,7674	SI	4,8544
6	Hall 2	94,1176	Hall 2	87,0588	SI	7,0588
7	Hall 2	96,4706	Sin coincid.	0	NO	96,4706
8	Hall 2	100	Pasillo	96,988	NO	3,012
9	Hall 2	99,3671	Pasillo	87,5	NO	11,8671
10	Hall 2	97,6471	Hall 2	82,3529	SI	15,2942
11	Hall 2	97,6471	Hall 2	91,8605	SI	5,7866
12	Hall 2	96,4706	Hall 2	91,3793	SI	5,0913
13	Hall 2	91,1877	Hall 2	86,3636	SI	4,8241
14	Hall 2	88,8889	Hall 2	86,4458	SI	2,4431
15	Hall 2	98,8372	Hall 2	90,6977	SI	8,1395

Hall 3

Tabla 21 Resultados simulación hall 3 con ruido

	Resultado SO	Fiab (%)	Resultado CO	Fiab (%)	Comp.	Diferencia
1	Hall 3	100	Hall 3	96,9512	SI	3,0488
2	Hall 3	100	Pasillo	81,1111	NO	18,8889
3	Hall 3	100	Hall 3	91,0112	SI	8,9888
4	Hall 3	98,9583	Hall 3	84,1463	SI	14,812
5	Hall 3	91,25	Hall 3	82,716	SI	8,534
6	Hall 3	88,25	Hall 3	86,3614	SI	1,8886

7	Hall 3	87,0588	Hall 3	80,102	SI	6,9568
8	Hall 3	96,4706	Hall 3	88,3721	SI	8,0985
9	Hall 3	96	Hall 3	92,3077	SI	3,6923
10	Hall 3	96,0843	Hall 3	89,1566	SI	6,9277
11	Hall 3	87,1429	Hall 3	81,3953	SI	5,7476
12	Hall 3	89,5798	Hall 3	81,3953	SI	8,1845
13	Hall 3	88,0899	Hall 3	83,4157	SI	4,6742
14	Hall 3	97,4684	Hall 3	93,0233	SI	4,4451
15	Hall 3	100	Sin coincid.	0	NO	100

Laboratorios

Tabla 22 Resultados simulación laboratorio con ruido

	Resultado SO	Fiab (%)	Resultado CO	Fiab (%)	Comp.	Diferencia
1	Laboratorio	100	Laboratorio	95,5556	SI	4,4444
2	Laboratorio	98,6667	Laboratorio	92,2222	SI	6,4445
3	Laboratorio	97,037	Laboratorio	86,6667	SI	10,3703
4	Laboratorio	98,6667	Laboratorio	85	SI	13,6667
5	Laboratorio	100	Laboratorio	90	SI	10
6	Laboratorio	97,1111	Laboratorio	95,5556	SI	1,5555
7	Laboratorio	90	Laboratorio	88,2222	SI	1,7778
8	Laboratorio	100	Laboratorio	100	SI	0
9	Laboratorio	100	Laboratorio	85,5556	SI	14,4444
10	Laboratorio	100	Laboratorio	95,5556	SI	4,4444
11	Laboratorio	90,5556	Pasillo	83,8889	NO	6,6667
12	Laboratorio	100	Laboratorio	95,5556	SI	4,4444
13	Laboratorio	100	Laboratorio	100	SI	0
14	Laboratorio	93,2584	Laboratorio	81,1111	SI	12,1473
15	Laboratorio	97,2222	Sin coincid.	0	NO	97,2222

Despacho

Tabla 23 Resultados simulación con ruido

	Resultado SO	Fiab (%)	Resultado CO	Fiab (%)	Comp.	Diferencia
1	Despacho	100	Despacho	91,8519	SI	8,1481

2	Despacho	94,4444	Despacho	88,8889	SI	5,5555
3	Despacho	95,5556	Despacho	97,2222	SI	-1,6666
4	Despacho	98,0556	Despacho	97,7778	SI	0,2778
5	Despacho	98,1481	Despacho	93,7037	SI	4,4444
6	Despacho	97,5	Despacho	97,3333	SI	0,1667
7	Despacho	98,8506	Despacho	94,4444	SI	4,4062
8	Despacho	95,1111	Despacho	91,8519	SI	3,2592
9	Despacho	94,0741	Despacho	87,2222	SI	6,8519
10	Despacho	99,6296	Despacho	90,6667	SI	8,9629
11	Despacho	100	Despacho	96,6667	SI	3,3333
12	Despacho	96,6667	Despacho	90,6667	SI	6
13	Despacho	97,7778	Despacho	96,1111	SI	1,6667
14	Despacho	90	Despacho	86,6667	SI	3,3333
15	Despacho	92,222	Despacho	94,4444	SI	-2,2224

Rellano

Tabla 24 Resultados simulación rellano con ruido

	Resultado SO	Fiab (%)	Resultado CO	Fiab (%)	Comp.	Diferencia
1	Rellano	100	Rellano	96,6667	SI	3,3333
2	Rellano	91,1111	Rellano	86,6667	SI	4,4444
3	Rellano	100	Rellano	95,5556	SI	4,4444
4	Rellano	100	Hall 3	90	NO	10
5	Rellano	88,5057	Sin coincid.	0	NO	88,5057
6	Rellano	100	Rellano	92,2222	SI	7,7778
7	Rellano	89,7727	Rellano	84,375	SI	5,3977
8	Rellano	100	Laboratorio	89,3574	NO	10,6426
9	Rellano	100	Rellano	95,625	SI	4,375
10	Rellano	95,1852	Rellano	91,1111	SI	4,0741
11	Rellano	100	Rellano	96,6667	SI	3,3333
12	Rellano	100	Rellano	95,5556	SI	4,4444
13	Rellano	100	Rellano	90	SI	10
14	Rellano	100	Rellano	95,0617	SI	4,9383
15	Rellano	98,9035	Rellano	85,2321	SI	13,6714

7.2 ANEXO B. LIBSVM: MANUAL DE USO

La librería Libsvm (2) es un conjunto de funciones en diferentes lenguajes de programación que desarrollan distintos algoritmos de máquinas de soporte vectorial. Fue realizada por Chih-Chung Chang and Chih-Jen Lin y resuelve problemas SVM de clasificación (C-SVM, nu-SVM), de regresión (épsilon-SVR, nu-SVR) y de estimación-distribución (SVM de una clase) tanto de clases binarias como de multiclase.

En este proyecto solo se emplearon los códigos en Matlab y de entre las funciones disponibles, se hizo uso de svmtrain y svmpredict.

7.2.1 SVMTRAIN

Es la función encargada de entrenar la SVM, se le introducen una serie de datos de entrada y genera un modelo en forma de variable tipo estructura que caracteriza nuestra SVM.

La librería Libsvm permite clasificar datos con etiquetas binarias (-1, 1), como en el caso de este proyecto, o problemas con mayor número de clases. En caso de emplear más de dos clases en el modelo de la SVM las etiquetas tendrán que ser positivas empezando por la número 1 (2, 3, 4 …).

La llamada a la función se realiza con la siguiente sintaxis:

```
model = svmtrain(training_label_vector, training_instance_matrix,
'libsvm_options')
```

- Model: es el resultado del entramiento. Una variable tipo estructura que define matemáticamente el modelo sobre el que se clasifican los futuros datos de entrada. Un ejemplo del modelo de un despacho sería:

  Parameters: [5x1 double]
  nr_class: 2
  totalSV: 213
  rho: 0.1410
  Label: [2x1 double]
  ProbA: []
  ProbB: []
  nSV: [2x1 double]
  sv_coef: [213x1 double]
  SVs: [213x2 double]

  Parameters es el vector que guarda el tipo de parámetros que definimos a la hora de ejecutar la opción. Serían los correspondientes al tipo de función SVM usada, el kernel, variables… se detallarán en el apartado7.2.1.1.

En este ejemplo tendríamos dos clases (nr_class = 2), siendo las etiquetas 0 y 1 (Label: 2x1). En total se generaron 213 vectores soporte, 105 para la clase 0 y 108 para la clase 1 (totalSV = 213, dentro de la variable nSV vienen detallados cuantos para cada clase). Rho es el término b de la ecuación $y_i(\omega^T x_i + b) \geq 1 - \xi_i$ de la SVM y SVs y sv_coef definen los vectores soporte calculados.

- training_label_vector: es el vector de dos columnas de datos de entrenamiento. Son datos que deben ser conocidos ya que tenemos que conocer su clase para poder entrenar a la máquina.
- training_instance_matriz: es la columna de etiquetas del vector de datos de entrenamiento. Debe tener una longitud igual a la del vector training_label_vector y definen la clase del vector de su correspondiente fila. Las etiquetas pueden ser binarias (-1, 1) o positivas para problemas con un número mayor de clases (1, 2, 3…)
- 'libsvm options': son las opciones de caracterización del modelo empleadas por la librería Libsvm,

7.2.1.1 OPCIONES

Las opciones se introducen entre comillas simples en la llamada a la función. El identificador de la función siempre va precedido de un guión '-' y el valor se escribe a continuación. La separación entre opciones es con un espacio simple.

Un ejemplo de la llamada a la función con opciones es:

```
model = svmtrain(training_label_vector, training_instance_matrix, '-
s 0 -t 1 -d 4')
```

Las distintas opciones de parametrización son:

- s: tipo de SVM. Por defecto viene definido el valor 0
  - 0. C-SVM. Para emplear una máquina de vector soporte tipo C o también llamada tipo 1. Este tipo de máquina de soporte vectorial permite que haya datos dentro del margen, con una holgura ξ_i mayor que 0. El parámetro C permite controlar el número de errores de clasificación permitidos en la etapa de aprendizaje. Cuanto mayor es C menos ejemplos de entrenamiento serán mal clasificados. La SVM de margen máximo se corresponde con el caso en el que $C = \infty$.
  - 1. NU-SVM. Para utilizar una SVM tipo 2 o NU. En este caso se mantiene la holgura ξ_i pero en vez del parámetro C se maneja el parámetro nu (ν) que también permite cierto control sobre el número de datos de entrenamiento y vectores soporte. Dicho parámetro ν ϵ (0, 1]
  - 2. One-class SVM. Este tipo de máquina de soporte vectorial emplea solo una única clase que no es necesario introducir durante el entrenamiento. El vector de entrenamiento debe ser de datos que

tengan esa única clase. Este modo es más complicado de configurar (a la hora de elegir los valores de un y gamma).

- 3. Epsilon-SVR. Esta opción permite utilizar máquinas de soporte vectorial tipo regresión (las anteriores son de tipo clasificación), en concreto la de tipo épsilon.
- 4. NU-SVR. Para el empleo de una máquina de soporte vectorial de regresión tipo NU.

- t: tipo de kernel. Las variantes que permite Libsvm son las que se desarrollaron en los capítulos 2 y 3. Por defecto viene seleccionada la opción 2, función gaussiana.
  - 0. Kernel lineal. Es igual al polinómico de grado 1. El único parámetro a configurar sería C o NU en función de la máquina de soporte vectorial elegida.

$$x^T \cdot x'$$

  - 1. Kernel polinómico. Una función asociada a un polinomio con coeficientes a_i de propiedad conmutativa. Según el grado del polinomio, representado en i, podrá ser una función lineal (grado1), cuadrática (grado 2), cúbica (grado 3)…

$$(\gamma \cdot x^T \cdot x' + coef)^d \qquad \gamma, coef \in \Re\,, \qquad d \in \chi$$

    Se pueden configurar los valores relativos a la SVM elegida (C), gamma, el término independiente y el grado del polinomio.

  - 2. Kernel gaussiano. La gráfica de la función es una campana (campana de Gauss) definida por la siguiente expresión:

$$\gamma \cdot exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right) \qquad \gamma \in \Re \qquad \sigma > 0$$

    Se pueden configurar los valores relativos a la SVM elegida (C), gamma y el valor de sigma.

  - 3. Kernel sigmoide. Es un tipo de función que modela muchos procesos naturales y curvas de aprendizaje. Su gráfica tiene forma de S, con un inicio lento, una aceleración intermedia y final igual de suave que el principio. El grupo de funciones sigmoide incluye tangentes parabólicas, arcotangentes, funciones logísticas…

$$\tanh(\gamma(x^T \cdot x') + coef) \qquad \gamma, coef \in \Re$$

    Se pueden configurar los valores relativos a la SVM elegida (C), gamma y el término independiente.

  - 4. Kernel precomputarizado.
- d. Grado. Sirve para definir el grado del polinomio empleado como kernel. Por defecto viene seleccionado el de tercer grado, 3. A mayor grado del polinomio mayor coste computacional al entrenar la SVM

- g. Gamma. Define el valor de gamma. Es un parámetro importante para ajustar los modelos. Su importancia depende del tipo de datos de entrenamientos. En algunos casos no variará a penas el modelo, mientras que en otros su valor resultará fundamental para el ajuste del mismo. Por defecto el valor de gamma es 1.
- r. Coeficiente. Aplica en el kernel polinómico y en el sigmoide. Por defecto su valor es 0.
- c. Coste. Parámetro de las máquinas de soporte vectorial de clasificación tipo 1 y de las SVM de regresión, tanto la épsilon como la NU. Es el que define el margen de dichos modelos. Al igual que el parámetro gamma, su valor predeterminado es 1.
- n. NU. Parámetro de las máquinas de soporte vectorial de clasificación tipo 2, de las SVM de una clase y de las nu-SVR. Su valor por defecto es 0,5
- p. Epsilon. Define el valor del parámetro en las epsilon-SVR. Por defecto es 0.1
- m. Tamaño de la caché. Es el valor en MB de la memoria caché que emplea la librería para sus cálculos. Por defecto es 100 MB. Un valor muy bajo puede no ser suficiente para el correcto desarrollo de Matlab, mientras que un valor muy alto, si el ordenador no está bien dimensionado, puede interferir con otros programas o con el propio sistema operativo.
- e. Epsilon. Establece el valor de tolerancia empleado como criterio de finalización de bucles y funciones empleados por la librería. Por defecto es 0.001.
- q. Parámetro empleado si no queremos que el programa muestre el resultado en pantalla (*quiet mode*)

7.2.2 SVMPREDICT

La función svmpredict es la encargada de ejecutar la clasificación basada en SVM. Necesita como punto de partida modelos creados a través de la función svmtrain y datos de entrada en el sistema (vectores de posición y etiqueta) y en función de estos inputs clasifica dichos datos.

La llamada a la función se realiza con la siguiente sintaxis:

```
[predicted_label, accuracy, decision_values/prob_estimates] =
svmpredict(testing_label_vector, testing_instance_matrix, model,
'libsvm_options')
```

Las salidas de la función son:

- predicted_label: es un vector columna de etiquetas correspondientes a los datos de entrada. Las calcula en función del modelo que le hemos suministrado comparando en qué punto del modelo está el vector de entrada y clasificándolo.
- accuracy: muestra la fiabilidad total de la clasificación. Como uno de los datos de entrada es la etiqueta de los datos, se compara la etiqueta de entrada con la obtenida de la clasificación y calcula el porcentaje de acierto. En este

proyecto todas las etiquetas de entrada que se introducen en la función son 1 (positiva, que el dato forma parte del modelo que se está comprobando), por lo que el valor de accuracy es la fiabilidad de dicha clasificación.
- decisión_values/pro_estimates: si seleccionamos esta variable obtenemos la probabilidad de estimación de los vectores. En este proyecto no la empleamos en las conclusiones.

Las entradas a la función svmpredict son:

- model: modelo creado con la función svmtrain. Debe estar en formato variable estructura de Matlab.
- testing_label_vector: vector columna con las etiquetas de los datos de entrada estimadas, que resulta útil para calcular la fiabilidad de la clasificación. En el presente proyecto este vector siempre tiene valor 1 para que la fiabilidad sea respecto al 100% de una clasificación positiva.
- testing_instance_matrix: vector de dos columnas que contiene los datos de entrada a la función. Estos datos son de posición en X e Y y son los que contrasta contra el modelo para clasificarlos.
- libsvm_options: las opciones descritas en el apartado 7.2.1.

7.2.3 SVMTOY

Svmtoy es una función desarrollada por Hsuan-Tien Lin, que permite la salida gráfica en dos dimensiones del problema de clasificación resuelto por svmpredict.

La función en primer lugar, partiendo de los datos de localización y etiquetas que se le proporcionan, llama a svmtrain y crea un modelo base. Posteriormente desarrolla una malla después de escalar los datos de entrada e introduce los datos de esa malla en la función svmpredict, creando una clasificación de todos los datos de la cuadrícula que posteriormente dibuja para generar la salida a pantalla de los datos iniciales con una serie de curvas de nivel que representan a los vectores soporte.

La llamada a la función se realiza con la siguiente sintaxis:

```
svmtoy(label_matrix, instance_matrix, options, contour_level)
```

Las entradas a la función son:

- label_matrix: vector columna con las etiquetas de los datos de entrada.
- instance_matrix: vector de dos columnas, posición en X e Y.
- options: las mismas que emplea Libsvm
- contour_level: por defecto [0 0]. Define el margen de los vectores soporte al representarlos. Para ampliar en +-1 el margen se cambiaría por [-1 0 1].